VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

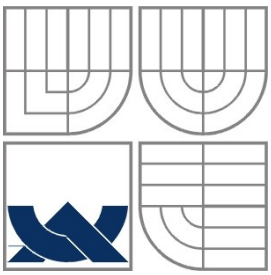
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

NÁSTROJ PRO ANALÝZU KOMUNIKACE HLASOVÝCH PŘENOSŮ PŘES H.323

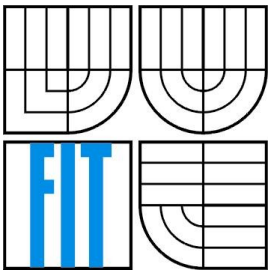
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE FILIP KARPÍŠEK
AUTHOR

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

NÁSTROJ PRO ANALÝZU KOMUNIKACE HLASOVÝCH PŘENOSŮ PŘES H.323

ANALYZING IP TELEPHONY OVER H.323

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

FILIP KARPÍŠEK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. PETR MATOUŠEK, Ph.D

BRNO 2012

Abstrakt

Tato práce popisuje protokoly obsažené ve standardu H.323. Zabývá se také nástroji použitelnými pro analýzu tohoto standardu. Hlavním cílem této práce je popsat návrh a implementaci nástroje pro analýzu komunikace hlasových přenosů právě přes H.323. Nástroj vyhledává a dekóduje signalizační zprávy, ze kterých získá jak informace o hovoru samotném (začátek a konec hovoru, telefonní čísla účastníků, atd.), tak informace nutné pro zachycení multimediálních dat. V případě zachycení multimediálních data tato exportuje ve formátu vhodném pro následné zpracování. Informace o hovorech jsou exportovány ve formátu XML.

Abstract

This thesis describes protocols from H.323 standard. Tools used for analyzing VoIP calls over H.323 are described. The main object is to describe a design and implementation of a tool for analyzing IP telephony over H.323. The tool seeks and decodes signaling messages. These messages are analyzed for information about call itself (time of beginning and end of a call, call numbers of participants, etc.) and for information necessary for capturing multimedia data. When captured, multimedia data are exported for proper post-processing. Call information are exported as XML structures.

Klíčová slova

H.323, PER, ASN.1, RTP, analýza VoIP

Keywords

H.323, PER, ASN.1, RTP, VoIP analysis

Citace

Filip Karpíšek: Nástroj pro analýzu komunikace hlasových přenosů přes H.323, bakalářská práce, Brno, FIT VUT v Brně, 2012

Nástroj pro analýzu komunikace hlasových přenosů přes H.323

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Petra Matouška, Ph.D.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Filip Karpíšek
Datum (3. květen 2012)

Poděkování

Tímto chci poděkovat Ing. Petrovi Matouškovi, Ph.D. jako vedoucímu mé bakalářské práce za jeho rady a odbornou pomoc, kterou mi při vedení této práce poskytl.

Obsah

1 Úvod

Telefon je v současné době stále velice hojně využívané zařízení. S postupem doby však přišla digitalizace, která umožňuje postupné nasazování levnějších komunikačních technologií sdílejících jedinou paketovou síť. Standardem pro multimediální komunikaci po paketových sítích je právě H.323 [1], který umožňuje kromě hlasu přenášet i video, umožňuje výstavbu škálovatelných řešení ať už pro poskytovatele telefonních služeb nebo pro soukromé subjekty.

Tato práce vznikla za podpory grantu MV a výzkumného záměru Moderní prostředky pro boj s kybernetickou kriminalitou na Internetu nové generace: VG20102015022.

1.1 Motivace

H.323 je poměrně složitý standard. Jeho úplná analýza není triviálním úkolem, neboť díky své škálovatelnosti poskytuje různé varianty ustavování hovorů. Využívá široké spektrum protokolů, které je nutné umět dekódovat a správně interpretovat.

V současné době však neexistuje nástroj, který by prováděl analýzu komunikace a poskytoval získané informace v požadovaném formátu. Dostupnými nástroji a jejich možnostmi se budeme více zabývat v kapitole 3.

Uplatněním úspěšné analýzy hovoru vedených přes H.323 může být využito například k účtování takových hovorů či sbírání statistik.

1.2 Cíl

Cílem této práce je podrobně prozkoumat možnosti standardu H.323 z hlediska jeho analýzy, navrhnout, implementovat a otestovat univerzální nástroj pro analýzu tohoto standardu.

Měl by vzniknout nástroj dostatečně robustní, čímž se rozumí zvládnutí úspěšné zpracování vybrané množiny scénářů, které H.323 díky své škálovatelnosti poskytuje.

Přínos takového nástroje pak spočívá v automatické transformaci síťového provozu na množinu uživatelsky užitečných informací, čímž se rozumí soubor zvukových stop hovoru a informací o něm – začátek a konec hovoru, počet signalizačních paketů, atd. Rovněž by mělo být možné jej integrovat do jiného systému, např. pro účtování hovorů či zpracování statistik.

1.3 Návrh řešení

Hovor v paketové síti skládá ze signalizace a samotný multimediálních dat (audio a/nebo video). Prvním krokem k úspěšnému řešení je tedy korektní dekódování signalizace hovorů. Z dekódované signalizace bude nutné získat informace pro získání dat z přenosu zvuku/videa (adresy a porty RTP toků, použité kodeky) a další užitečné informace (telefonní čísla, další identifikátory účastníků hovoru, procentuální podíl zachycených audio dat).

Dále bude nutné takto získané informace propojit. Například pro získání délky hovoru je nutné identifikovat zprávy zahajující hovor a zprávy ten samý hovor ukončující.

Posledním krokem je export takto získaných hodnot, aby bylo možné je dále zpracovávat nezávisle na tomto nástroji. Informace o hovoru budou exportovány do XML struktury, audio data pak budeme exportovat přímo a pokud to bude možné, tak i jako soubory ve formátu, který lze bez větších potíží přehrát (wav, mp3, ogg, …).

Samotnou práci na vývoji jsme tedy rozdělili do tří částí. Nejprve jsme v laboratoři vytvořili sadu souborů s hovory přes H.323, které jsme analyzovali pomocí nástroje Wireshark spolu se studiem standardů. Následoval návrh aplikace s využitím knihoven a prostředí jazyka Python. Nakonec jsme nástroj testovali na výše zmíněných laboratorních datech a na datech z reálného provozu.

Tato práce je rozvržena následujícím způsobem. V kapitole 2 rozebíráme protokoly standardu H.323 z pohledu analýzy. V kapitole 3 uvádíme současné nástroje pro analýzu H.323, jejich vlastnosti a možnosti. Návrh a implementaci nástroje jsme popsali v kapitole 4, způsob a výsledky testování uvádíme v kapitole 5.

2 Komunikace podle standardu H.323

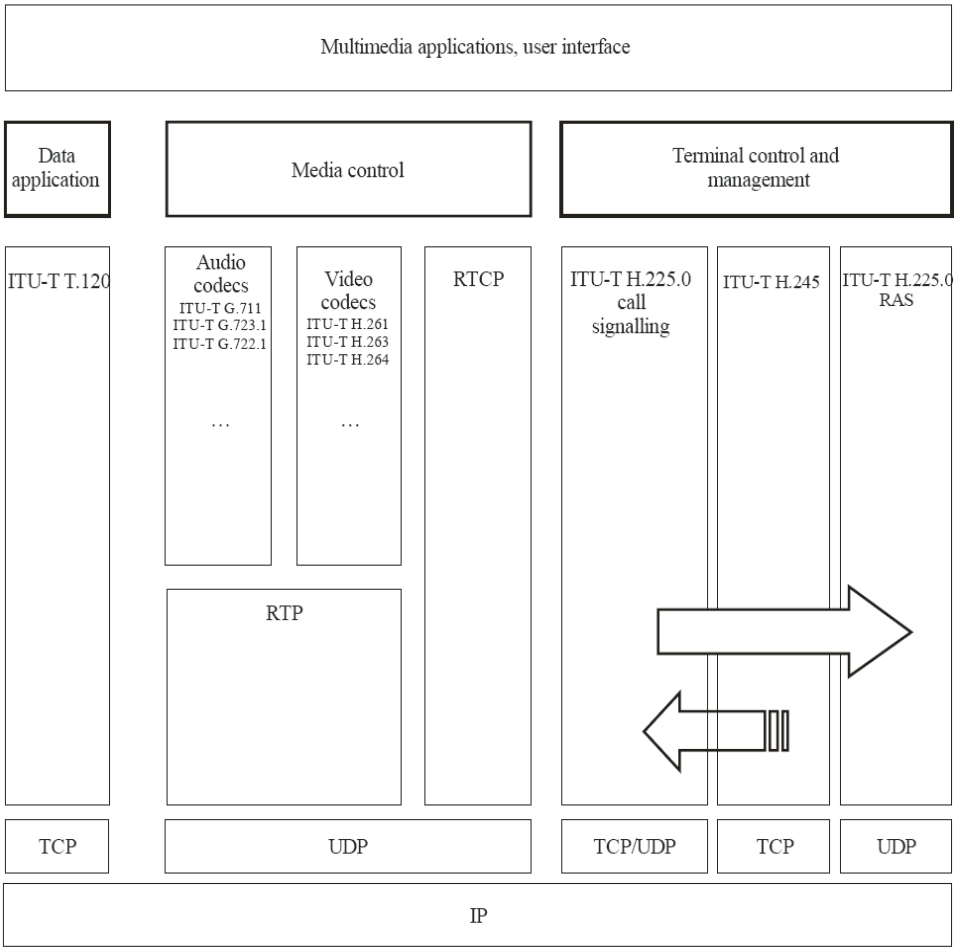
V této kapitole popíšeme komunikaci podle standardu H.323. Seznámíme se s použitými protokoly a různými scénáři ustavení hovoru, abychom je mohli všechny úspěšně analyzovat.

H.323 je doporučení Mezinárodní telekomunikační unie (ITU) definující protokoly pro audio-vizuální komunikaci v paketové síti. Toto doporučení má za cíl standardizovat tyto protokoly a architekturu za účelem univerzálnosti použití. Taková univerzálnost spočívá ve škálovatelnosti řešení a velkém množství volitelných položek, které jsou použity jen v případě potřeby. Rovněž obsahuje definice bezpečnosti.

Z pohledu analýzy komunikace je podstatná struktura a formát signalizačních zpráv. Zprávy jsou uloženy ve formátu struktur definovaných Abstraktní syntaktickou notací (ASN.1 [2]) kódovanými Pravidly zhuštěného kódování (PER [3]). Dále je ještě nutné uvažovat zapouzdření těchto zpráv do protokolu Q.931 [7], které jsou ještě dále zapouzdřeny do zpráv TPKT [6].

Pro přenos multimediálního obsahu je použit protokol RTP [8].

Na obrázku 2.1 jsou uvedeny protokoly a standardy, které jsou součástí H.323.



Obrázek 2.1: Protokoly a standardy obsažené v H.323 [1, str. 298]

2.1 Kódování PER (Packed Encoding Rules)

Jedná se o velice efektivní binární kódování kódující struktury ASN.. Umožňuje kódovat se zarovnáním na byty (varianta ALIGNED), nebo bez zarovnání (varianta UNALIGNED). H.323 využívá zarovnané varianty.

PER o poznání náročnější na dekódování, například v porovnání s BER. BER totiž u každé položky explicitně uvádí typ a délku položky, PER neuvádí typ nikdy, délku pouze tehdy, pokud je proměnlivá. Je tedy nutné znát přesně strukturu dekódované zprávy, jinak není možné jakékoliv dekódování. Podrobnosti je možno dohledat v [5].

Jelikož je délka i přítomnost některých položek neznámá, je nutné pro získání hodnoty určité položky zpracovat zprávu právě až po hledanou položku, není tedy možné ve zprávě přímo vyhledávat.

Pro toto kódování (na rozdíl od BER) neexistuje nástroj pro dekódování, neboť je nutná znalost přesné struktury dekódované zprávy. Následuje ukázka kódování této struktury (převzato z [3], str.44):

```
PersonnelRecord ::= [APPLICATION 0] IMPLICIT SET {
  name           Name,
  title          [0] VisibleString,
  number         EmployeeNumber,
  dateOfHire     [1] Date,
  nameOfSpouse   [2] Name,
  children       [3] IMPLICIT
    SEQUENCE OF ChildInformation DEFAULT {} }

ChildInformation ::= SET {
  name           Name,
  dateOfBirth    [0] Date}

Name ::= [APPLICATION 1] IMPLICIT SEQUENCE {
  givenName      VisibleString,
  initial        VisibleString,
  familyName     VisibleString}

EmployeeNumber ::= [APPLICATION 2] IMPLICIT INTEGER

Date ::= [APPLICATION 3] IMPLICIT VisibleString – YYYYMMDD
```

Struktura bude zakódována s těmito hodnotami:

```
{ name {givenName "John",initial "P",familyName "Smith"},
 title                  "Director",
 number                 51,
 dateOfHire             "19710917",
 nameOfSpouse
 {givenName "Mary",initial "T",familyName "Smith"},
 children
 {{name {givenName "Ralph",initial "T",familyName "Smith"},
dateOfBirth "19571111"},
 {name {givenName "Susan",initial "B",familyName "Jones"},
 dateOfBirth "19590717"}}}
```

Výsledek zakódování:

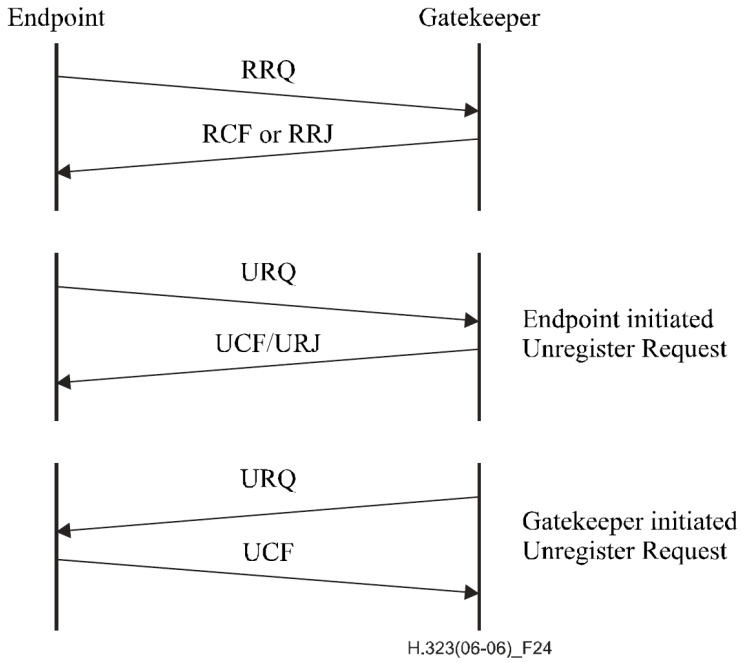
```
1xxxxxxx                                                                    1 indicates "children" is present

00000100                                                                    Length of name.givenName = 4
01001010 01101111 01101000 01101110                                         name.givenName = "John"

00000001                                                                    Length of name.initial = 1
01010000                                                                    name.initial = "P"

00000101                                                                    Length of name.familyName = 5
01010011 01101101 01101001 01110100 01101000                                name.familyName = "Smith"

00000001                                                                    Length of (employee) number = 1
00110011                                                                    (employee) number = 51

00001000                                                                    Length of title = 8
01000100 01101001 01110010 01100101 01100011 01110100 01101111 01110010   title = "Director"

00001000                                                                    Length of dateOfHire = 8
00110001 00111001 00110111 00110001 00110000 00111001 00110001 00110111   dateOfHire = "19710917"

00000100                                                                    Length of nameOfSpouse.givenName = 4
01001101 01100001 01110010 01111001                                         nameOfSpouse.givenName = "Mary"

00000001                                                                    Length of nameOfSpouse.initial = 1
01010100                                                                    nameOfSpouse.initial = "T"

00000101                                                                    Length of nameOfSpouse.familyName = 5
01010011 01101101 01101001 01110100 01101000                                nameOfSpouse.familyName = "Smith"

00000010                                                                    Number of children = 2

00000101                                                                    Length of children[0].givenName = 5
01010010 01100001 01101100 01110000 01101000                                children[0].givenName = "Ralph"

00000001                                                                    Length of children[0].initial = 1
01010100                                                                    children[0].initial = "T"

00000101                                                                    Length of children[0].familyName = 5
01010011 01101101 01101001 01110100 01101000                                children[0].familyName = "Smith"

00001000                                                                    Length of children[0].dateOfBirth = 8
00110001 00111001 00110101 00110111 00110001 00110001 00110001 00110001   children[0].dateOfBirth =
"19571111"

00000101                                                                    Length of children[1].givenName = 5
01010011 01110101 01110011 01100001 01101110                                children[1].givenName = "Susan"

00000001                                                                    Length of children[1].initial = 1
01000010                                                                    children[1].initial = "B"

00000101                                                                    Length of children[1].familyName = 5
01001010 01101111 01101110 01100101 01110011                                children[1].familyName = "Jones"

00001000                                                                    Length of children[1].dateOfBirth = 8
00110001 00111001 00110101 00111001 00110000 00110111 00110001 00110111   children[1].dateOfBirth =
"19590717"
```

2.2 Protokol H.225, RAS (Register, Admision, Status)

Jedná se o signalizační zprávy pro doplňkové služby: vyhledání ústředny (GRQ, GCF, GRJ), přihlášení/odhlášení na ústřednu (RRQ, RCF, RRJ), vyhledání volané strany (ARQ, ACF, ARJ) a další. Tyto zprávy se převážně vyskytují mezi ústřednou a koncovým zařízením (terminálem).

Z hlediska analýzy provozu je nutné tyto zprávy sledovat, neboť při přihlášení na ústřednu může terminál uvést jiný, než standardní port pro příjem signalizačních zpráv. Stejná situace nastává při vytváření hovoru, kdy je volaná strana registrovaná a přijímá signalizační zprávy na jiném portu. V tomto případě získá volající strana tuto informaci před začátkem procesu ustavování hovoru v odpovědi na vyhledání volané strany.

Na obrázku 2.2 jsou zobrazeny tři různé scénáře komunikace koncového zařízení s ústřednou týkající se registrace a odregistrace koncového zařízení. Při registraci lze zachytit identifikátor uživatele, který používá k registraci. Navíc může koncové zařízení specifikovat nestandardní port, na kterém očekává signalizační zprávy CS, takže je nutné proces registrace sledovat, aby bylo možné zpracovávat signalizaci na těchto neznámých portech.



Obrázek 2.2: Komunikace s ústřednou – registrace, odregistrace [1, str.55]

2.3 Protokol H.225, CS (Call Signaling)

Signalizační zprávy tohoto typu slouží k samotnému ustavení hovorů. Slouží k vyjednání parametrů spojení. Pokud je použita technologie FastStart, je vyjednávání rychlejší, neboť signalizační zprávy obsahují zapouzdřené zprávy protokolu H.245.

Bez technologie FastStart slouží tyto signalizační zprávy pouze pro vyjednání kanálu pro protokol H.245, které jsou pak směrovány na jiné porty než signalizační zprávy. Takové ustavení hovoru může mít dvě fáze.

2.4 Protokol RTP

Protokol RTP je určen pro přenos multimediálních dat, v případě VoIP komunikace nese hlasová případně obrazová data. Jeho detekce však není jednoduchá, protože je přenášen v protokolu UDP, který neobsahuje identifikátor protokolu vyšší vrstvy. Porty, na kterých je přenášen, jsou navíc vybírány dynamicky, což činí jeho detekci ještě komplikovanější.

2.5 Protokol H.245

Zprávy tohoto protokolu slouží k vyjednání parametrů kanálu samotného multimediálního toku, ať už je to audio či video. Podstatné jsou zejména adresy a porty RTP toků, případně použité kodeky.

Často je využita technologie FastStart, kdy jsou tyto zprávy zapouzdřeny do zpráv CS.

2.6 Role ústředny

Ústředna provádí v architektuře H.323 propojování koncových zařízení (a jiných ústředen) a služby překladu adres (např. z telefonních čísel na IP adresy). V některých případech může řídit konferenční hovory.

Základní funkce ústředny v H.323 architektuře je správa uživatelů a mapování identifikátorů (např. uživatelských jmen nebo telefonních čísel) na síťové adresy. Pro tuto funkcionalitu je nutná i komunikace s dalšími ústřednami, neboť je často nutné propojit terminály registrované (připojené) na různých ústřednách.

Režim běhu ústředny může umožňovat buď přímou signalizaci, nebo signalizaci řízenou přes ústřednu. Při žádosti o hovor (*ARQ*) od koncového zařízení dostane v odpovědi (*ACF*) v případě přímé signalizace adresu volaného koncového bodu. V případě signalizace přes ústřednu dostane adresu ústředny, terminál tedy nemá na výběr a musí zasílat signalizační zprávy přes ústřednu,

protože nezná adresu volané strany. RTP toky jsou v obou případech směrovány přímo mezi koncovými body bez účasti ústředny.

Obrázek 2.3 popisuje signalizaci vedenou přímo s použitím technologie FastStart. Význam jednotlivých zkratek a účel zpráv je následující:

1 **ARQ** (Admission Request) – žádost o spojení, obsahuje identifikátor volané strany (telefonní číslo, uživatelské jméno, …)

2 **ACF** (Admission Confirm) – potvrzení spojení, obsahuje IP adresu a port, kde volaná strana (Endpoint 2) očekává signalizační zprávy

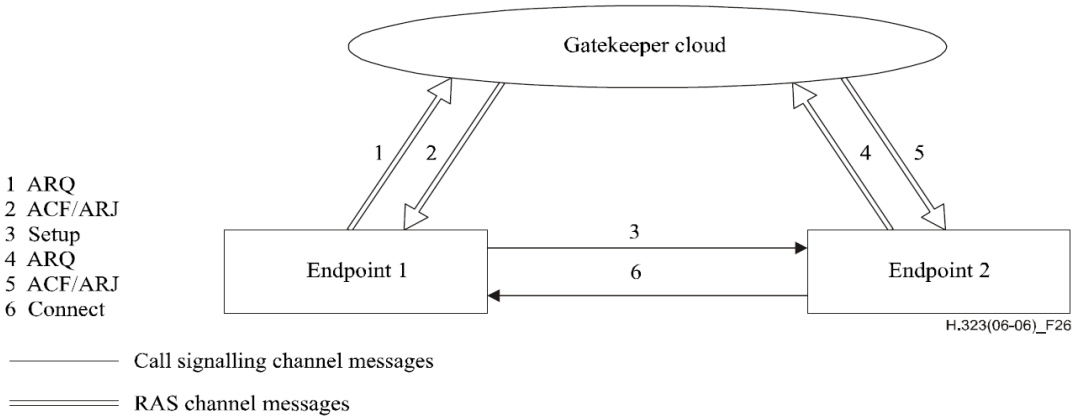
- **ARJ** (Admission Reject) – odmítnutí spojení, například z důvodu, že ústředna nezná volanou stranu (Endpoint 2), volající (Endpoint 1) není na ústředně zaregistrován, ústředna nemá zdroje k uskutečnění hovoru aj.

3 **Setup** – první zpráva určená pro volanou stranu (Endpoint 2), obsahuje informace nutné k ustavení hovoru, např. adresu a port jedné strany multimediálního RTP toku, seznam podporovaných kodeků, identifikátor volajícího (Endpoint 1) aj.

4 **ARQ** – žádost o povolení spojení

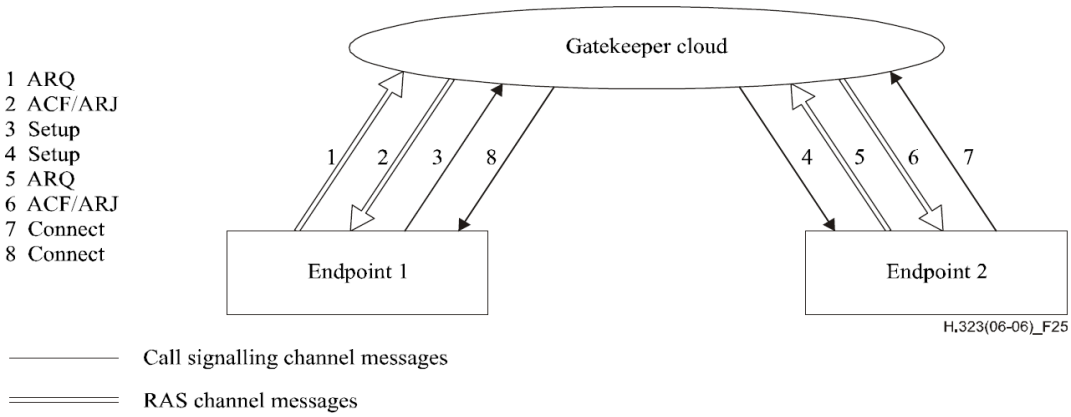
5 **ACF/ARJ** – povolení nebo zamítnutí spojení

6 **Connect** – poslední zpráva signalizační části ustavení hovoru, následuje multimediální RTP provoz



Obrázek 2.3: Signalizace vedená přímo mezi terminály [1, str.68]

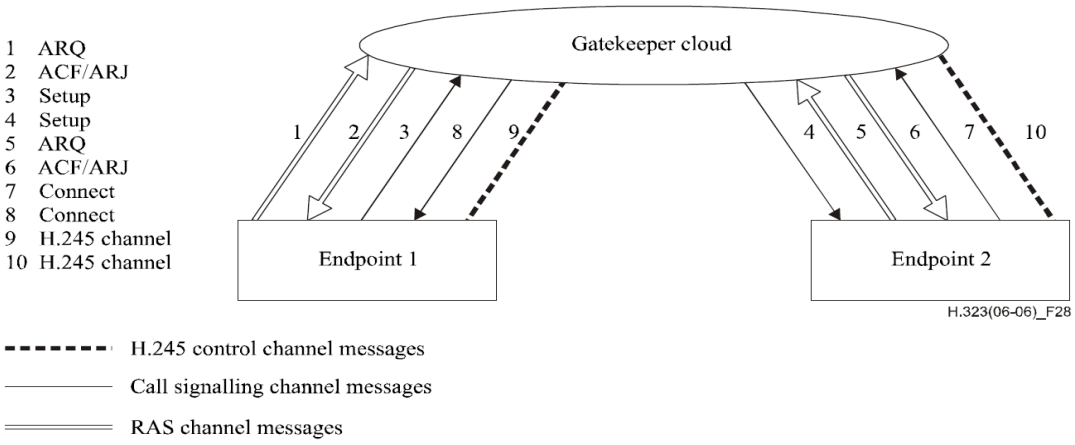
Obrázek 2.4 ukazuje průběh a cesty signalizace vedené přes ústřednu (nepřímá signalizace). Hlavní rozdíl spočívá v tom, že zpráva *ACF* neobsahuje adresu volané strany (Endpoint 2), ale samotné ústředny, čímž je vynuceno zasílání H.225 CS zpráv přes ústřednu. Ústředna tak může zasahovat do hodnot v signalizačních zprávách a upravovat je dle potřeby.



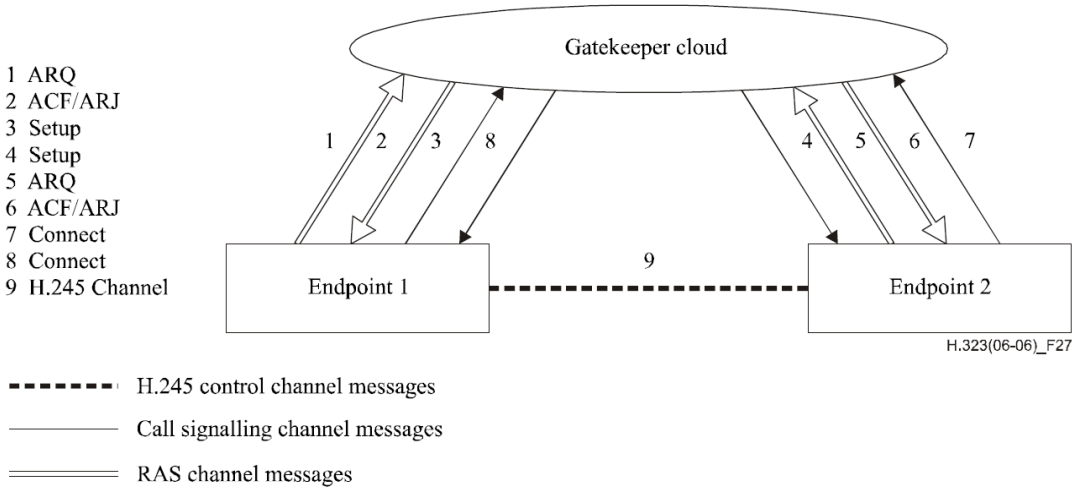
Obrázek 2.4: Signalizace přes ústřednu[1, str.68]

Obrázky 2.3 a 2.4 ukazují signalizaci s použitím technologie FastStart. Pokud není tato technologie použita (H.245 zprávy jsou posílány samostatně), existují v případě signalizace přes ústřednu dvě varianty směrování H.245 zpráv.

Obrázek 2.5 popisuje směrování H.245 zpráv přes ústřednu, obrázek 2.6 pak směrování přímé. Význam a obsah zpráv je stejný jako v předchozích dvou případech. Přibyla však signalizace H.245, ve které jsou předávány parametry toků RTP.



Obrázek 2.5: Směrování H.245 přes ústřednu[1, str.69]

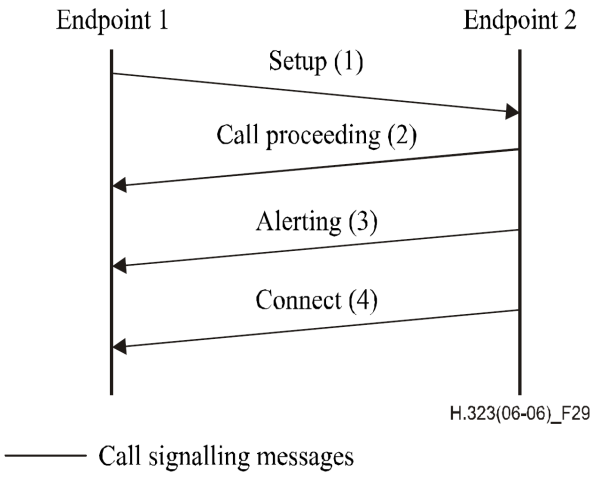


Obrázek 2.6: Přímý kanál H.245[1, str.69]

2.7 Scénáře ustavení hovoru

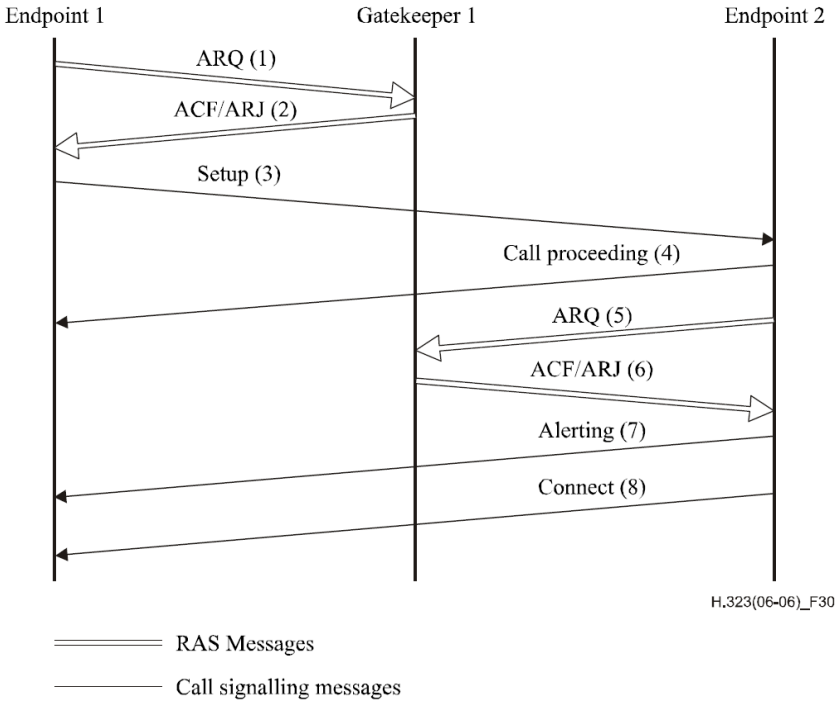
Ustavování hovoru může probíhat různými způsoby v závislosti na tom, zda jsou terminály registrované a v jakém režim běží ústředna. Z pohledu analýzy je důležité znát různé možnosti ustavení hovoru, aby je bylo možné korektně zpracovat.

Na obrázku 2.7 je průběh ustavení hovoru, kdy není ani jeden z terminálů registrovaný na ústředně a komunikace probíhá přímo. Volající zařízení (Endpoint 1) posílá volanému zařízení (Endpoint 2) na jeho známý signalizační port zprávu *Setup*(1) obsahující informaci o adrese a portu H.245 kanálu (případně adresu a port toku RTP a dostupné kodeky, pokud je využita technologie FastStart). Volané zařízení (Endpoint 1) odpovídá zprávou *Connect*(4), která obsahuje informaci o druhé straně kanálu H.245 (nebo o druhé straně toku RTP). Tímto je hovor ustaven a dochází k přenosu multimediálních dat v tocích RTP.



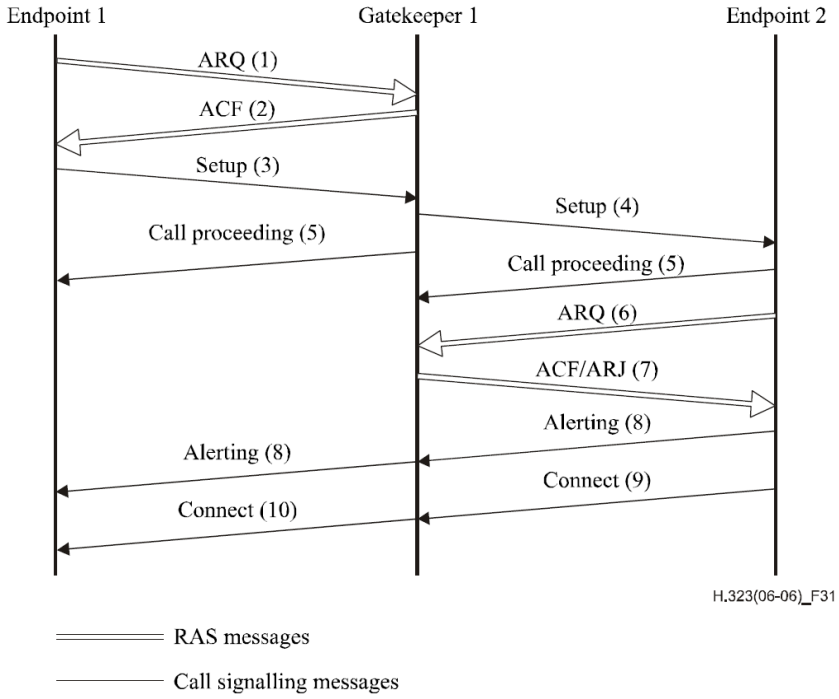
Obrázek 2.7: Základní ustavení hovoru bez ústředny [1, str.82]

Obrázek 2.8 ukazuje scénář, kdy jsou oba terminály registrované u stejné ústředny, která funguje v režimu přímé signalizace. Volající strana (Endpoint 1) zahajuje výměnou *ARQ*(1)/*ACF*(2) zpráv. Ústředna poskytne volající straně (Endpoint 1) adresu volané strany (Endpoint 2) včetně portu, který může být nestandardní. Na tuto adresu pak volající strana (Endpoint 1) posílá zprávu *Setup*(3) stejně jako v případě obrázku 2.7. Pokud si volaná strana (Endpoint 2) přeje přijmout hovor, iniciuje taktéž výměnu *ARQ*(5)/*ACF*(6) zpráv a odesílá zprávu *Call Proceeding*(4). Pokud však volaná strana (Endpoint 2) přijme z ústředny zprávu *ARJ*, odesílá volající straně (Endpoint 1) zprávu *Release Complete* a hovor musí být ustaven jinak. V případě povolení hovoru následuje zpráva *Connect*(8).



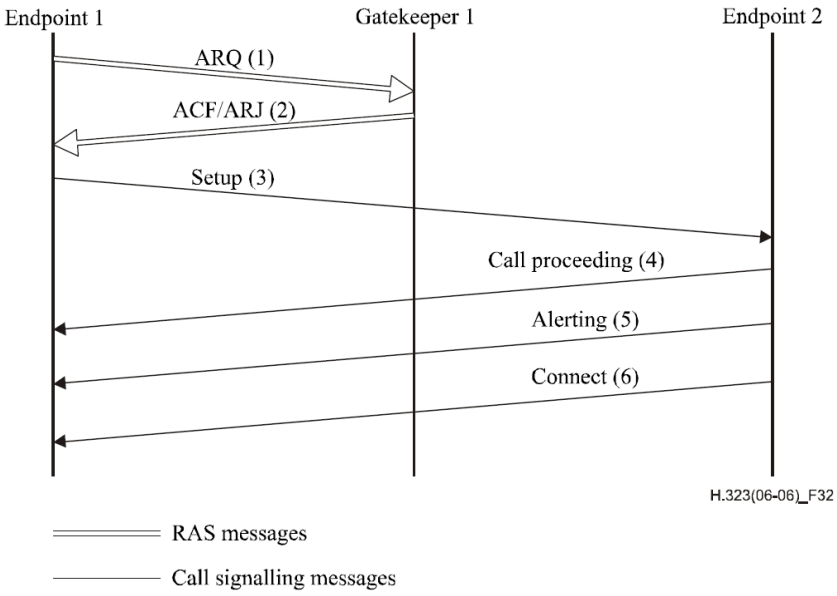
Obrázek 2.8: Ustavení hovoru, oba terminály jsou registrované, přímý režim [1, str.83]

Dalším případem je scénář na obrázku 2.9, kdy jsou oba terminály registrované u jedné ústředny, tak však běží v režimu signalizace přes ústřednu. Zprávy jsou odesílány ve stejném pořadí, ve zprávě *ACF*(2) však není obsažena adresa volané strany (Endpoint 2), ale adresa ústředny.



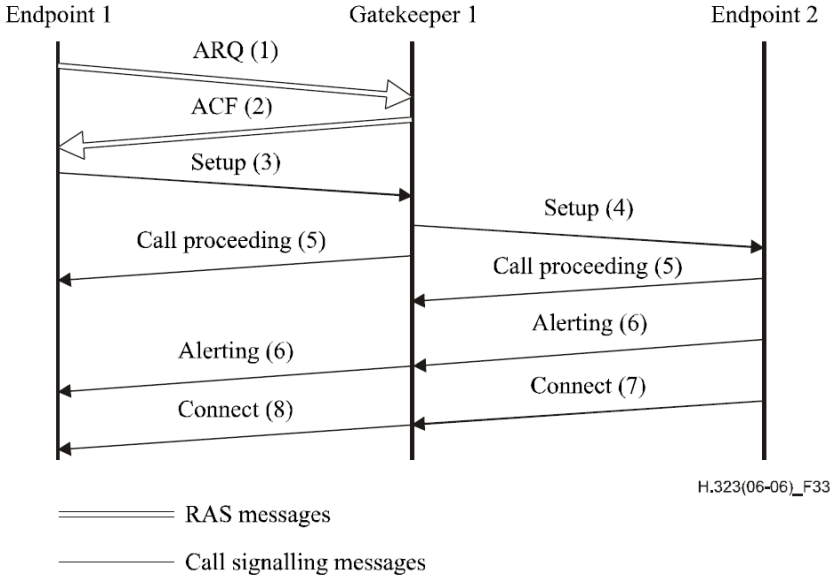
Obrázek 2.9: Oba terminály registrované u stejné ústředny, režimu nepřímý [1, str.84]

Další možností je, že je registrovaná pouze volající strana (Endpoint 1). Takový průběh ustavení hovoru je uveden na obrázku 2.10. Ustavení má podobný průběh, jako ustavení z obrázku 2.8, volaná strana (Endpoint 2) však neiniciuje výměnu zpráv *ARQ*/*ACF*.



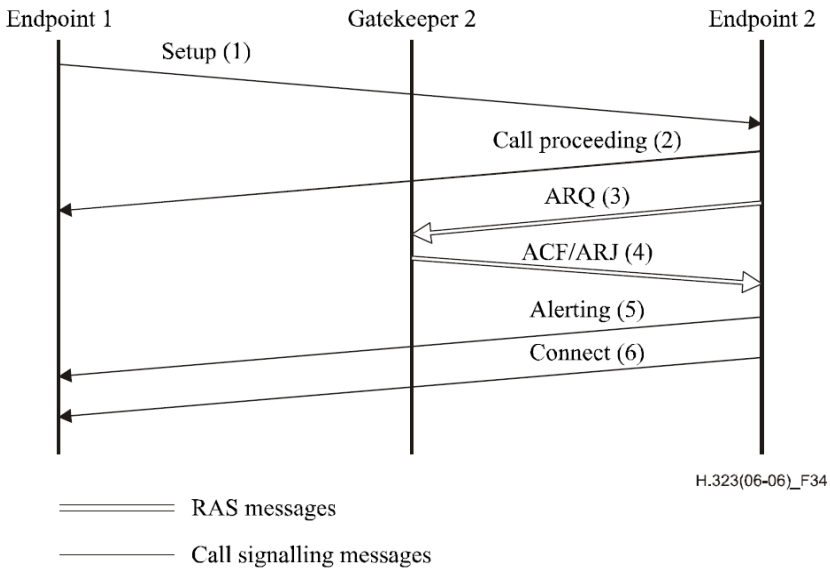
Obrázek 2.10: Volající terminál registrovaný, přímá signalizace [1, str.84]

Pokud zvolí ústředna režim nepřímé signalizace, ustavení hovoru probíhá jako na obrázku 2.11. Průběh je stejný, jako v předchozím případě, veškerá signalizace jde přes ústřednu.



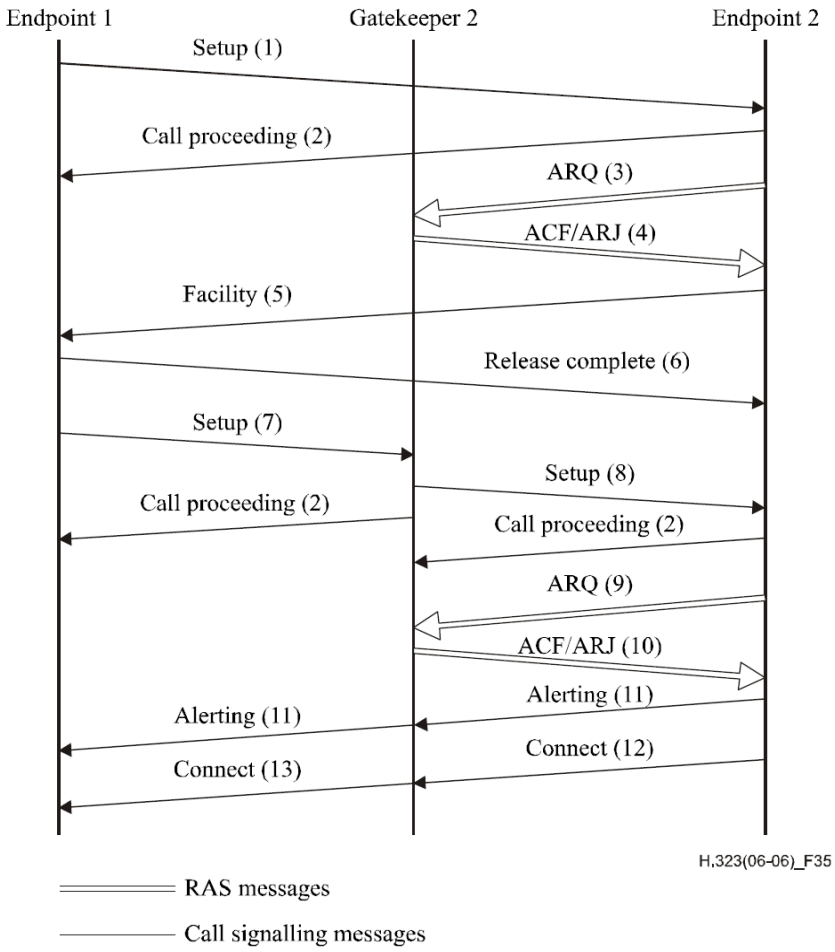
Obrázek 2.11: Volající terminál je registrovaný, nepřímá signalizace [1, str.85]

V opačném případě, kdy je registrovaný volaný terminál (Endpoint 2), probíhá ustavení podle obrázku 2.12. Zde však výměnu *ARQ*/*ACF* neprovádí volající strana (Endpoint 1).



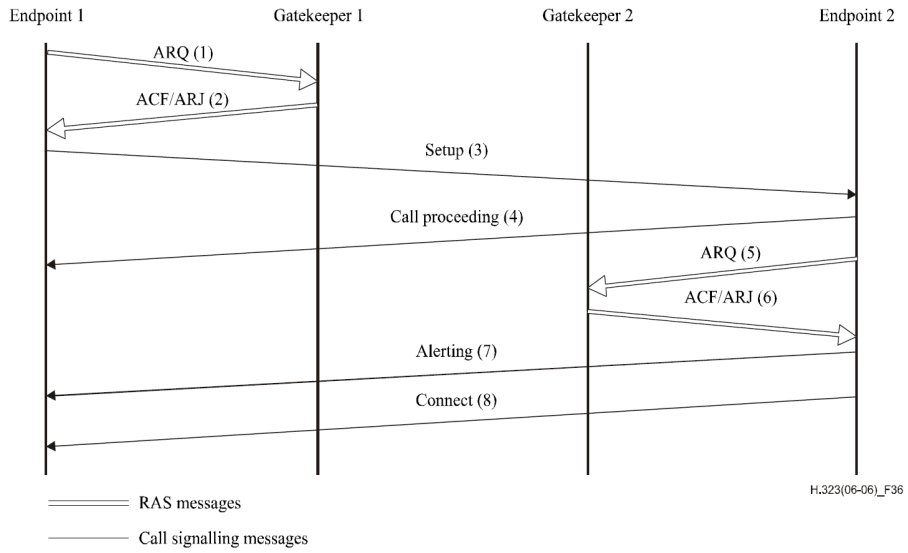
Obrázek 2.12: Volaný terminál je registrovaný, přímá signalizace [1, str.86]

V případě, kdy zvolí ústředna režim nepřímé signalizace, je scénář ustavení poněkud složitější, viz obrázek 2.13. Ústředna totiž donutí druhý terminál ukončit tím, že mu jako odpověď na zprávu *ARQ*(3) zašle zprávu *ARJ*(4) s kódem důvodu odmítnutí *routeCallToGatekeeper*. Druhý terminál reaguje tak, že zašle zprávu *Facility*(5) obsahující adresu a port, na kterém jeho ústředna přijímá zprávy CS. První terminál v tomto okamžiku ukončí pokus o ustavení přímým způsobem a vyjednává spojení přes ústřednu podobně jako ve scénáři na obrázku 2.9.



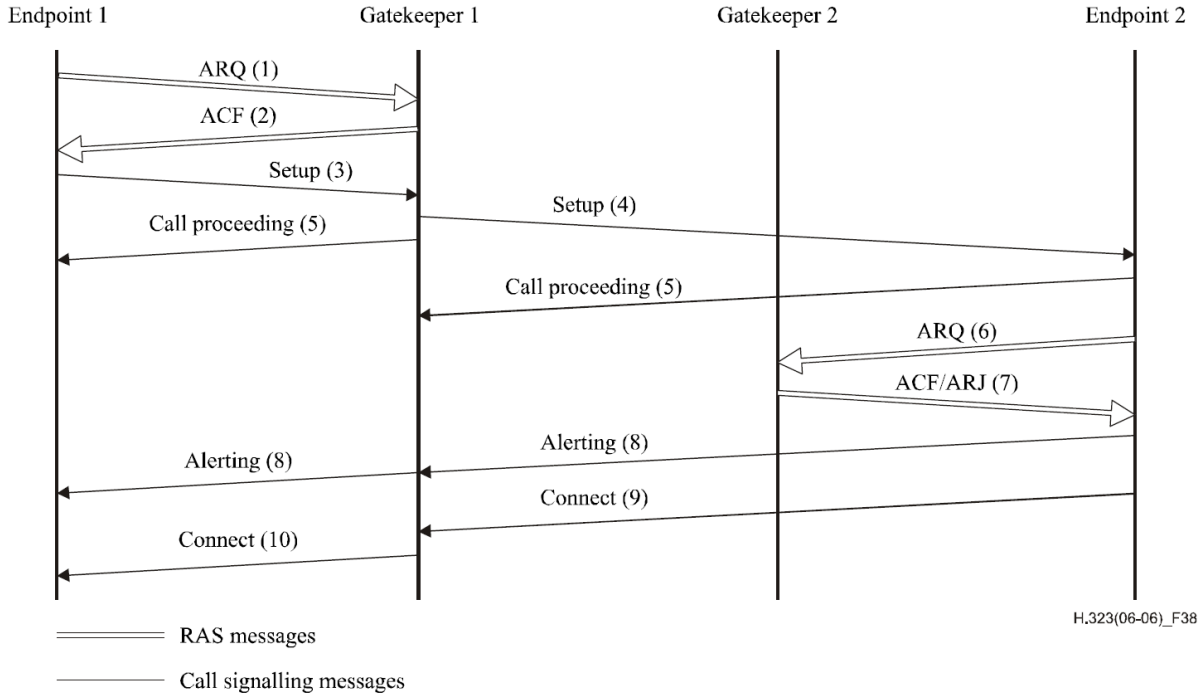
Obrázek 2.13: Volaný terminál je registrovaný, nepřímá signalizace [1, str.87]

Dalším případem je stav, kdy jsou oba terminály registrovány, tentokrát však na různých ústřednách. Obrázek 2.14 ukazuje průběh ustavování hovoru při přímé signalizaci na obou ústřednách. Postup ustavování je stejný jako ve scénáři na obrázku 2.9. Každé z koncových zařízení komunikuje se svou bránou, která mu dodá informace potřebné pro připojení ke vzdálenému zařízení. Ústředny mají metody pro komunikaci mezi sebou navzájem.



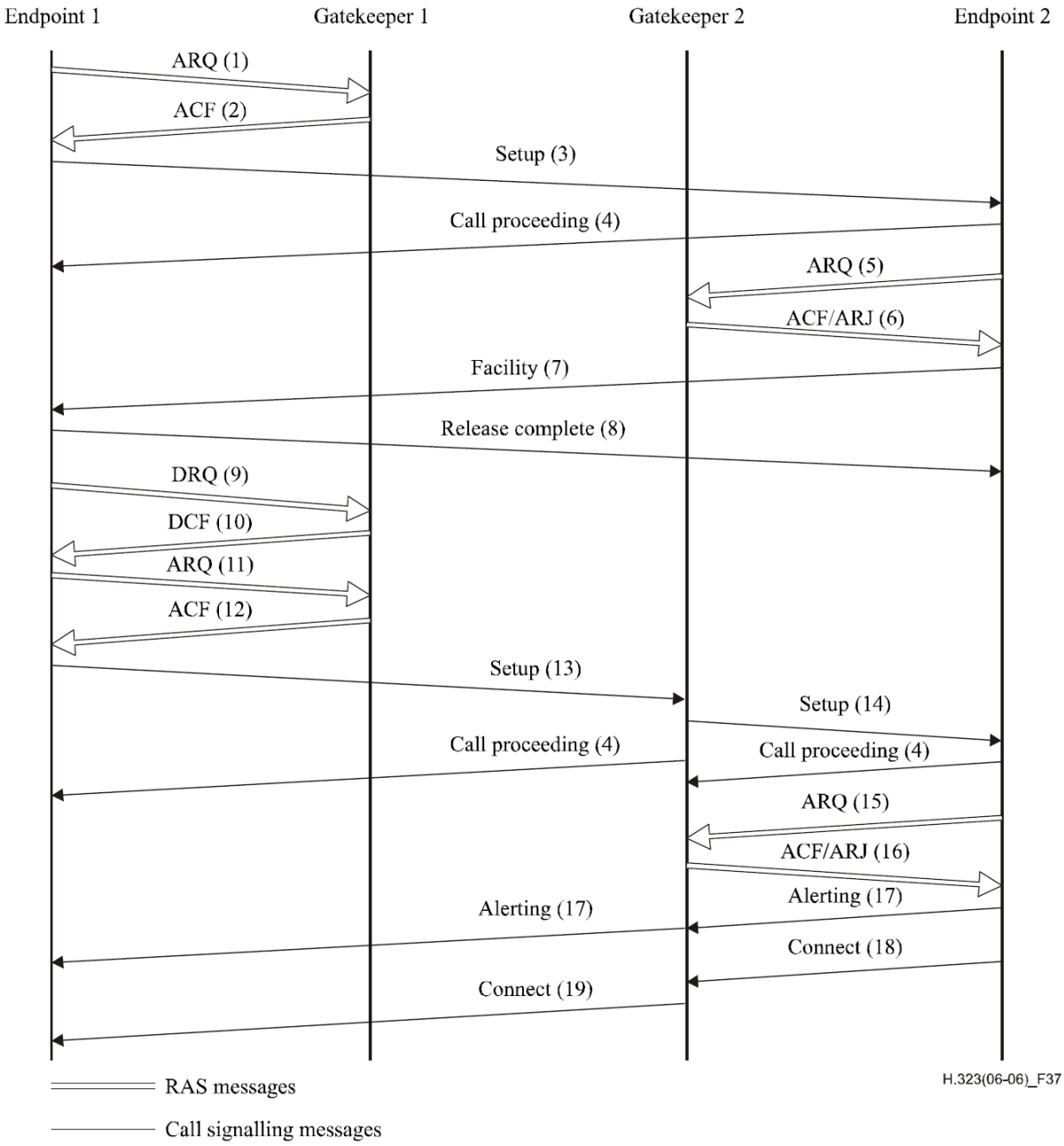
Obrázek 2.14: Oba terminály registrované, přímá signalizace u obou ústředen [1, str.88]

Opačný scénář, kdy jsou registrovány oba terminály, ale režim signalizace je opačný (nepřímá/přímá), je zobrazen na obrázku 2.15. Volající zařízení (Endpoint 1) se chová stejně, jako ve scénářích na obrázcích 2.9 a 2.11. Volané zařízen (Endpoint 2) se zase chová jako ve scénářích na obrázcích 2.8, 2.12 a 2.14. Ústředna na straně volajícího (Gatekeeper 1) se pak chová jako koncové zařízení.

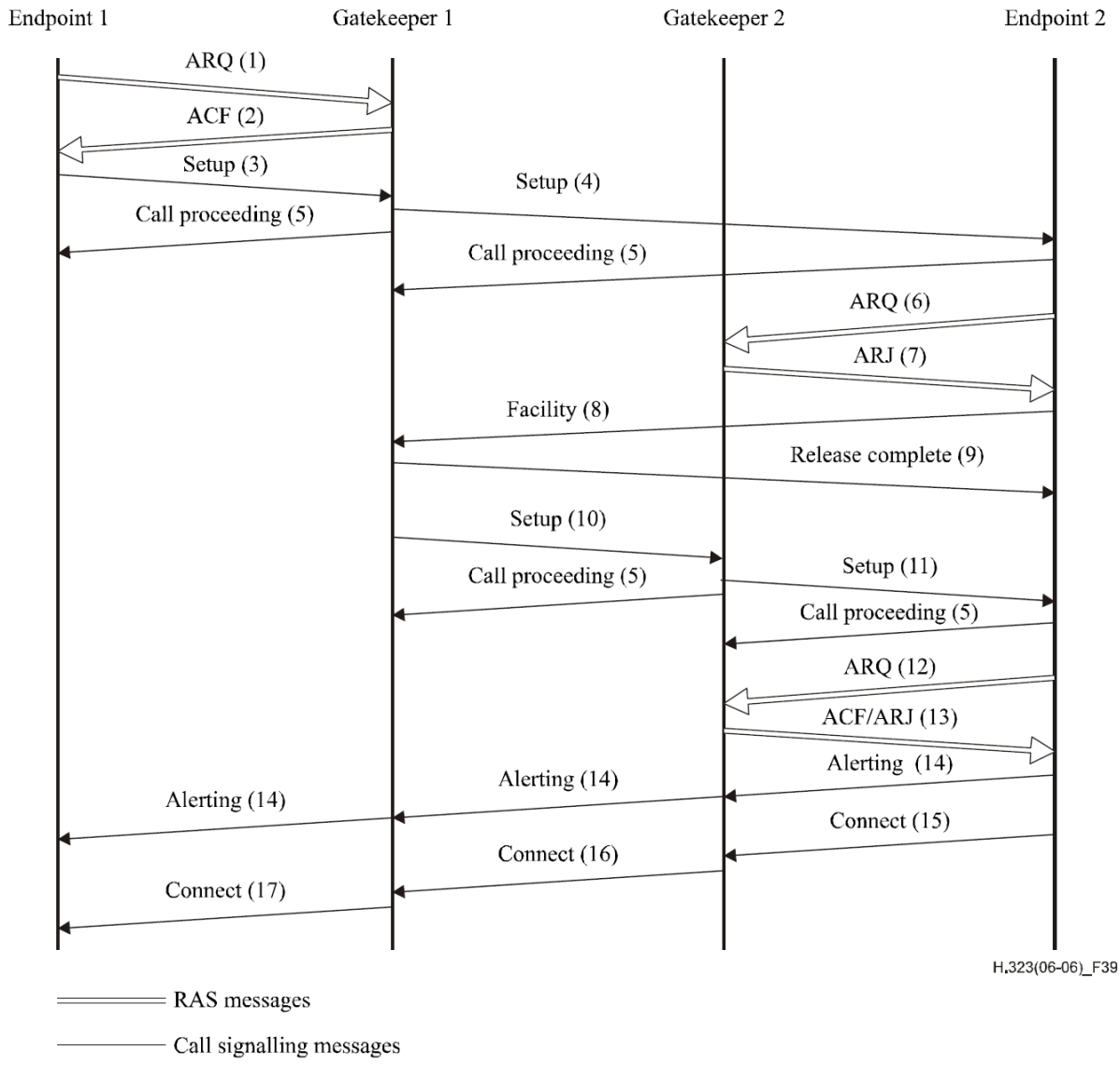


Obrázek 2.15: Oba terminály registrované, nepřímá/přímá signalizace [1, str.90]

Obrázek 2.16 ukazuje scénář ustavené hovoru, kdy jsou oba terminály registrované, první ústředna je v přímém režimu, druhá v režimu nepřímém. Scénář je nápadně podobný scénáři na obrázku 2.13 (registrovaný volaný, nepřímá signalizace). Volající zařízení (Endpoint 1) však navíc komunikuje se svou ústřednou a poprvé se objevuje výměna zpráv *DRQ*(9)/*DCF*(10), které ústředně oznamují, že došlo k ukončení hovoru.



Obrázek 2.16: Oba terminály registrované, přímá/nepřímá signalizace [1, str.89]



Obrázek 2.17: Oba terminály registrované, nepřímá/nepřímá signalizace [1, str.91]

Nejsložitější scénář nastává v případě, že obě ústředny vyžadují nepřímý režim signalizace, viz obrázek 2.17. Takový průběh je opět velice podobný průběhu na obrázku 2.13 (registrovaný volaný, nepřímá signalizace), první ústředna pak vystupuje v roli neregistrovaného volajícího.

Obecně lze tedy říci, že ve všech scénářích hovorů se vyskytují zprávy *Setup* a *Connect*, které obsahují informace stěžejní pro analýzu (adresy a porty toků RTP). Zprávy *Call Proceeding* a *Alerting* však také mohou obsahovat tyto informace, je tedy nutné zpracovávat i je, aby byla analýza robustnější, neboť může dojít k situaci, že nebude analyzovaný vzorek z nějakého důvodu obsahovat zprávy *Setup* či *Connect*.

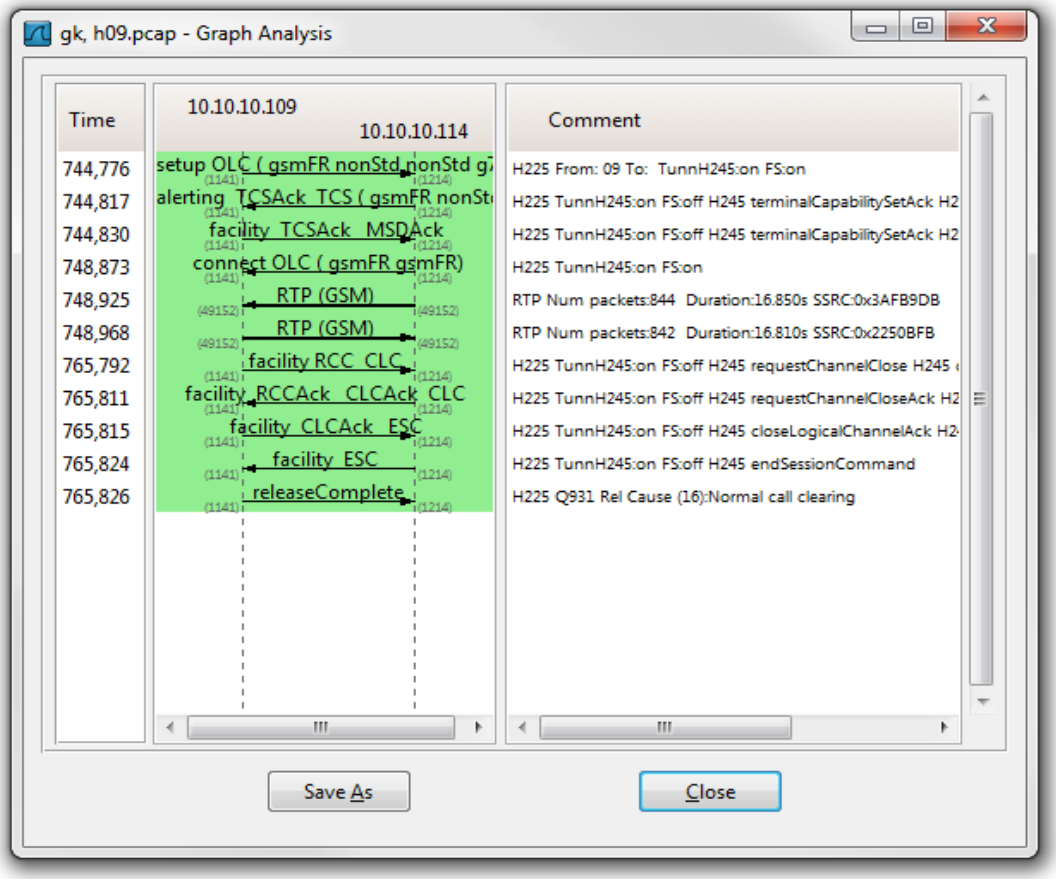
Bude rovněž nutné analyzovat zprávy signalizace *RAS*, neboť signalizace *Call Setup* nemusí být vždy zasílána na standardních portech. Informace, na kterém portu se signalizace *CS* objeví je právě ve zprávách signalizace *RAS*, konkrétně ve zprávách *ACF, RRQ* a *RCF*.

3 Současný stav, dostupné nástroje

Tato kapitola popisuje současný stav na poli analýzy hovorů vedených přes H.323. Popisuje převážně dostupné nástroje z hlediska analýzy hovorů VoIP. Konkrétně budou rozebrány nástroje Wireshark a Cain & Abel

3.1 Nástroj Wireshark

Wireshark[1] je univerzální open-source paketový analyzátor, který kromě jiného umožňuje i analýzu VoIP obecně, H.323 nevyjímaje. Umožňuje jak pohled na hovory, tak podrobnou analýzu jednotlivých paketů, přičemž signalizační zprávy jsou zaneseny do flow grafu. Příklad takového flow grafu je na obrázku 3.1.



Obrázek 3.1: Flow graf ustavení hovoru, Wireshark

Jeho největší výhodou je možnost přístupu ke každému jednotlivému paketu a podrobná analýza obsahu.

1 **http://www.wireshark.org/**

Nevýhodou pak zůstává nemožnost hromadně pracovat s hovory (nelze filtrovat, pouze řadit, a to pouze podle některých položek) a nemožnost hromadného exportu. Rovněž je nutné pro každý hovor ručně zjišťovat veškeré podstatné informace – začátek, konec, délka, procentuální poměr zachyceného provozu atp.

V některých případech dochází ke špatné interpretaci signalizace. Například hovor je ve stavu CANCELLED, protože se v souboru k danému hovoru nacházejí pouze zprávy *Setup* a *ReleaseComplete*, které mezi sebou však mají rozteč osmi minut.

Další nevýhodou je pak nemožnost vyhledávat pakety podle některých položek, například *callIdentifier*.

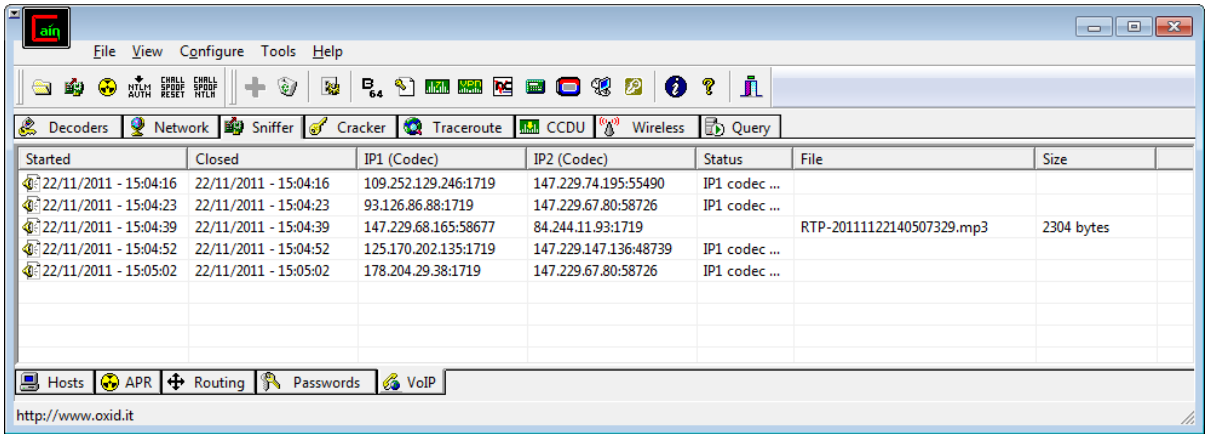
3.2 Nástroj Cain & Abel

Dalším částečně použitelným nástrojem pro transformaci síťového provozu na hlasové stopy a metadata je Cain & Abel[2]. Jeho primárním zaměřením je testování bezpečnosti sítí a síťových aplikací. Umožňuje získávat nezabezpečená uživatelská jména a hesla ze síťové komunikace, případně z prostředí operačního systému.

Co se VoIP týče, dokáže C&A z úspěšně zachycených hovorů exportovat hlasová data v běžně přehratelném formátu (wav, mp3).

O těchto hovorech však poskytuje velmi málo informací (adresy a porty toků RTP, kodek). Není možné hovory žádným způsobem identifikovat (pouze podle adres), řadit nebo filtrovat. Jakýkoliv export metadat o hovorech rovněž není podporován.

Na obrázku 3.2 je ukázka zpracovaných hovorů.



Obrázek 3.2: Cain & Abel

Žádný z dostupných nástrojů neumožňuje hromadný export informací o hovorech a není možné je zapojit do jiného systému, což je rovněž účelem této práce.

2 **http://www.oxid.it/cain.html**

4 Návrh a implementace nástroje

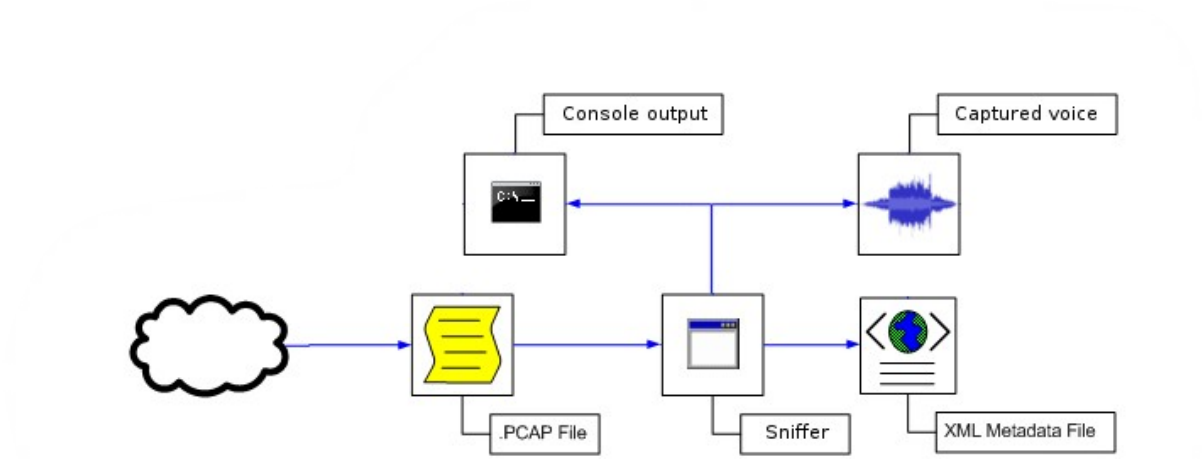
V této kapitole popíšeme účel a koncepce nástroje, jeho vnitřní architektura, funkcionalita jednotlivých částí, programové a implementační prostředí a ovládání programu.

Účelem nástroje je transformace vstupního souboru síťových dat na informace o hovorech, případně doplnění těchto informací o multimediální data daných hovorů.

4.1 Vstupy a výstupy

Na obrázku 4.1 je znázorněny vstupy a výstupy systému. Jediným vstupem je síťový provoz, ať už je reprezentován souborem se zachycenými pakety ve formátu pcap, nebo zachytáván na síťovém rozhraní.

Výstupem jsou pak metadata o hovorech, soubory se zachyceným zvukem a informační výpisy na konzoli.



Obrázek 4.1: Vstupy a výstupy nástroje

Na obrázku 4.2 je ukázka vstupního souboru, obrázku 4.3 pak náhled souboru XML reprezentujícího stejný vstupní soubor.

No.	Time	Source	Destination	Protocol	Info
1810	2011-07-30 16:59:24.065723	10.0.0.3	10.0.0.2	H.225.0/H.245	CS: setup OpenLogicalC
1811	2011-07-30 16:59:24.256658	10.0.0.2	10.0.0.3	H.225.0/H.245	CS: alerting terminalC
1813	2011-07-30 16:59:24.298811	10.0.0.3	10.0.0.2	H.225.0/H.245	CS: facility terminalC
1818	2011-07-30 16:59:29.260294	10.0.0.2	10.0.0.3	H.225.0	CS: connect OpenLogic
4975	2011-07-30 17:00:02.839250	10.0.0.3	10.0.0.2	H.225.0/H.245	CS: facility requestCl
4976	2011-07-30 17:00:02.847985	10.0.0.2	10.0.0.3	H.225.0/H.245	CS: facility requestCl
4978	2011-07-30 17:00:02.867631	10.0.0.3	10.0.0.2	H.225.0/H.245	CS: facility closeLog
4980	2011-07-30 17:00:02.900692	10.0.0.2	10.0.0.3	H.225.0/H.245	CS: facility endSessi
4982	2011-07-30 17:00:02.915413	10.0.0.3	10.0.0.2	H.225.0	CS: releaseComplete

Obrázek 4.2: Ukázka vstupního souboru

```
<?xml version="1.0" encoding="UTF-8"?>
<log>
  <user guid="10.0.0.3:1621">
    <protocol val="H.323">
      <event type="call">
        <callid>95041fad404e2c4193af9adec673e15c</callid>
        <timestamp type="begin">Jul 30, 2011 16:59:24</timestamp>
        <timestamp type="end">Jul 30, 2011 17:00:02</timestamp>
        <src>10.0.0.3:49152</src>
        <dst>10.0.0.2:49152</dst>
        <content codec="G.711,A-law(8)" destination="10.0.0.2:49152" id="0"
          source="10.0.0.3:49152" type="RTP stream">
          export/hovor-real.pcap/call_0/10.0.0.3;49152-10.0.0.2;49152....raw
        </content>
        <content codec="G.711,A-law(8)" destination="10.0.0.3:49152" id="1"
          source="10.0.0.2:49152" type="RTP stream">
          export/hovor-real.pcap/call_0/10.0.0.2;49152-10.0.0.3;49152....raw
        </content>
      </event>
    </protocol>
  </user>
</log>
```

Obrázek 4.3: Ukázka výstupního XML souboru – informace o hovoru

4.2 Architektura

Systém je navržen objektově, jednotlivé části různým způsobem přispívají ke zpracování vstupního proudu paketů, ať už je zdrojem pcap soubor nebo síťové rozhraní. Na obrázku 4.4 je znázorněn způsob, jakým jsou jednotlivé pakety zpracovávány při průchodu systémem.

Pro přístup k síťové vrstvě používáme knihovny pcapy. Pro základní klasifikaci paketu je využito pcap filtru (viz manuálové stránky *man pcap-filter*), který je podle potřeby upravován a vybírá pouze pakety, které jsou pro analýzu důležité. Jeho výchozí hodnota při spuštění programu je nastavena tak, že filtr propouští pakety s čísly portů (zdrojových nebo cílových) 1718, 1719 a 1720, což jsou doporučené porty pro zprávy RAS a CS.

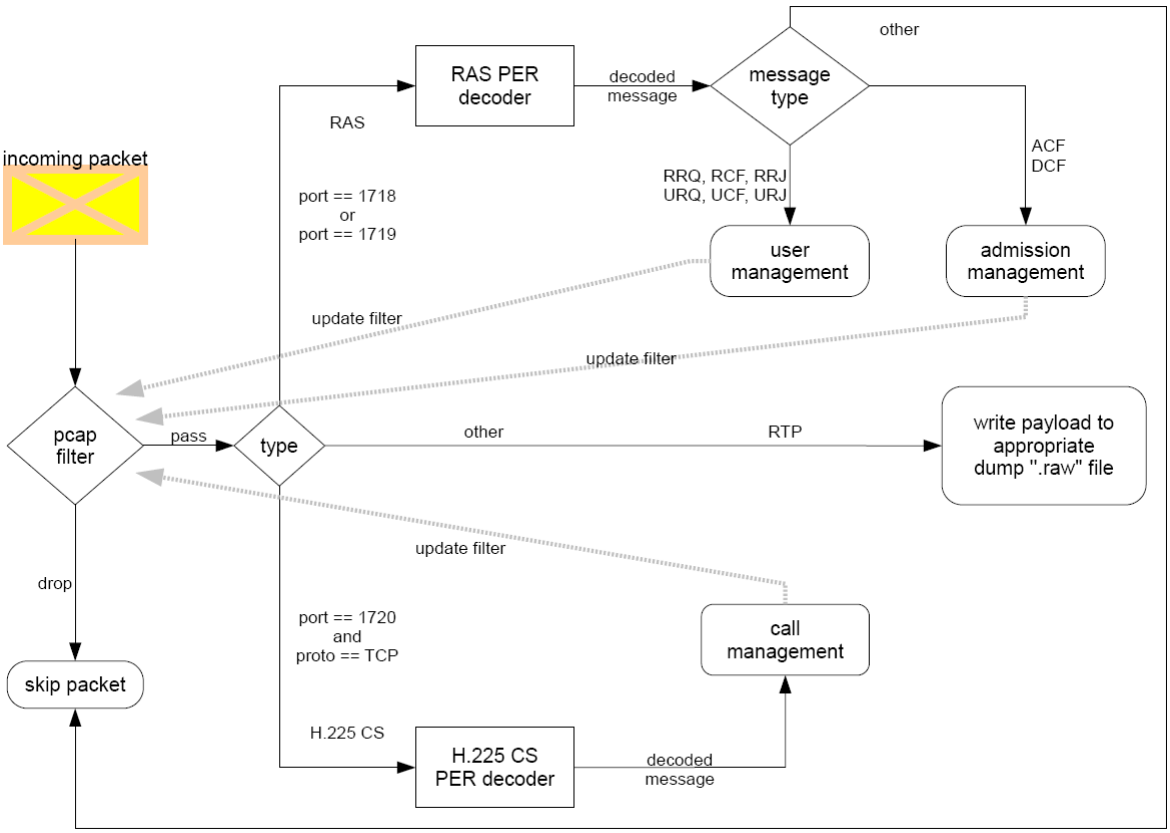
Po průchodu paketu filtrovacím pravidlem je určen typ daného paketu podle Tabulky 4.1.

Typ paketu	Protokol transportní vrstvy	Porty
H.225 RAS	UDP	1718,1719
H.225 CS	TCP	1720
RTP	UDP	dynamické

Tabulka 4.1: Typ paketu v závislosti na transportním protokolu a portech

Následně jsou zprávy RAS a CS rozkódovány a podle jejich obsahu jsou podle potřeby měněna filtrační pravidla. Například po detekci příchodu hovoru (získáno ze zpráv CS) jsou přidána pravidla pro průchod paketů RTP a ve správě hovorů je vytvořen objekt reprezentující daný hovor.

Po ukončení hovoru jsou tato pravidla opět odebrána a daný objekt zrušen. Podobně je ze signalizace RAS možno zjistit registrace a od-registrace uživatelů a žádosti o vytvoření hovoru, které mohou v některých případech rovněž vyžadovat přidání či ubrání filtračních pravidel.



Obrázek 4.4: Diagram zpracování paketu

Při příchodu paketů RTP je vyhledán hovor, ke kterému daný proud RTP náleží a data z paketu jsou zapsána do příslušného výstupní .raw souboru.

Při zpracování tedy dochází k následujícím operacím:

1. dekódování zpráv H.225 RAS a CS (PER, ASN.1)
2. výběr metadat o hovorech, viz. Tabulka 4.2 s ukázkovým obsahem záznamu o hovoru
3. práce s paketovým filtrem (přidávání a ubírání filtračních pravidel)
4. ukládání dat a prezentace

položka	hodnota
callidentifier	95041fad404e2c4193af9adec673e15c
začátek hovoru	Jul 30, 2011 16:59:29
konec hovoru	Jul 30, 2011 17:00:02
počet zpráv CS	9
forward IP adresa toku RTP	10.0.0.2:49152
reverse IP adresa toku RTP	10.0.0.3:49152

Tabulka 4.2: Ukázka záznamu hovoru

4.3 Komponenty

V této části budou popsány jednotlivé součásti systému a jejich funkcionalita a vzájemná interakce.

4.3.1 Načítání pcap souboru

K tomuto účelu je využito knihovny *pcapy*[3], která umožňuje načítat data jak ze souboru (offline analýza) tak ze síťového rozhraní (online nebo též live analýza). Způsob zpracování je následující:

1. příslušným způsobem je otevřen vstup (soubor nebo síťové zařízení)
2. před samotným zpracováním je vhodné nastavit filtr
3. ve smyčce je pro každý paket volána funkce, která ho zpracuje

Následuje krátká ukázka kódu v jazyce Python, která nahraje knihovnu pcapy, otevře příslušný vstup a nastaví funkci, která zpracovává pakety:

```
import pcapy
  ⋮
def process_packet(header, data):
     # funkce pro zpracovani paketu
     return
  ⋮
if(online):
     pc = pcapy.open_live(network_if, max_bytes, promiscuous, read_timeout)
if(offline):
     pc = pcapy.open_offline(file)
pc.setfilter(filter_configuration)
pc.loop(packet_limit, process_packet)
  ⋮
```

3 **http://oss.coresecurity.com/projects/pcapy.html**

4.3.2 Úprava dynamického filtru

Knihovna pcapy umožňuje filtrovat pakety podle určitých pravidel (shoda adresy, portu, protokolu síťové či transportní vrstvy apod.), která jsou v textové formě. Při spuštění nástroje je výchozí konfigurace následující:

```
port 1718 or port 1719 or port 1720
```

Filtr tedy propouští všechny pakety, jejichž hodnota cílového nebo zdrojového portu je v rozmezí 1718 až 1720. Po detekci hovoru z obrázku 4.2 dochází ke změně filtračních pravidel a to tak, že jsou přidány adresy a porty, které umožní průchod paketů RTP daného hovoru. Konfigurace filtru po přijetí zprávy *Setup* je pak následující:

```
port 1718 or port 1719 or port 1720 or (ip proto \udp and ((src 10.0.0.3 and
port 49152)or(dst 10.0.0.3 and port 49152)))
```

Pro jeden hovor pak může dojít k více změnám filtru, neboť během zpracovávání zpráv může docházet k upřesnění informací. Zpráva *Setup* totiž obsahuje pouze informace o jedné straně toku RTP. Po získání druhé strany toku se filtr změní do následujícího tvaru:

```
port 1718 or port 1719 or port 1720 or (ip proto \udp and (((src 10.0.0.2 and
port 49152) and (dst 10.0.0.3 and port 49152)) or ((src 10.0.0.3 and port 49152)
and (dst 10.0.0.2 and port 49152))))
```

Dojde tak k výraznému upřesnění filtračního pravidla, což vede na kvalitnější zpracování toků RTP, neboť klesá pravděpodobnost průniku nežádoucích paketů.

Po zpracování zprávy *Release Complete* dochází k odstranění přidaného pravidla a filtr je opět ve výchozím stavu.

Filtr je uložen v textovém formátu jako řetězec. Při potřebě úpravy dochází k jeho vytvoření spojením filtračních pravidel jednotlivých hovorů.

4.3.3 Dekodér H.323 signalizačních zpráv (PER)

Tato komponenta přijímá na svém vstupu pakety označené jako RAS či CS a dekóduje jejich obsah. Získanými informacemi poté naplňuje strukturu s informacemi o paketu.

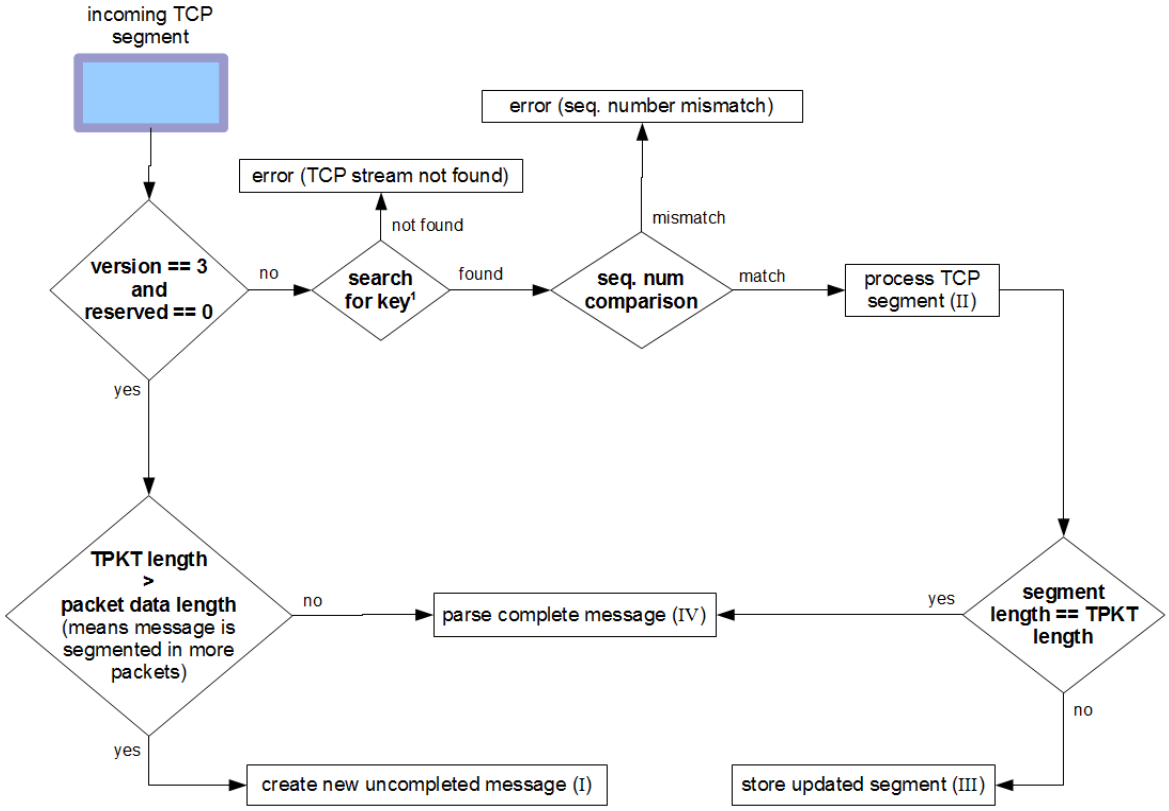
V případě potřeby dochází ke spojování segmentů TCP, a to v části dekódování TPKT, neboť TPKT obsahuje informaci o délce zprávy, je tedy možné zjistit, zda je zpráva přijata kompletně, nebo bude pokračovat v dalších paketech. Způsob spojování segmentů TCP je znázorněn na obrázku 4.5.

Nejprve jsou dekódovány všechny položky TPKT (version (dále V), reserved (dále R) i length). Pokud je V=3 a R=0, pak se pravděpodobně jedná o začátek zprávy. Pokud je očekávaná

délka větší, než skutečná délka, je zpráva nekompletní a je nutné vytvořit segment (I), který je uložen pro pozdější zpracování do databáze. Pokud je zpráva kompletní, může být rovnou zpracována.

Pokud je V různé od 3 nebo je R různé od 0, pak se pravděpodobně jedná o navazující segment. Pokud je nalezen odpovídající rozpracovaný segment a aktuální sekvenční číslo odpovídá číslu očekávanému, dojde ke zpracování segmentu (II), neboť se jedná o navazující segment.

Poté je provedena porovnání délky segmentu s délkou zprávy (získané v procesu I) a pokud se tyto položky rovnají, je zpráva kompletní a je předána ke zpracování, daný segment je pak z databáze odstraněn. Pokud se tyto položky nerovnají, zpráva ještě není kompletní a aktualizovaný segment je uložen zpět do databáze.



Obrázek 4.5: Diagram zpracování segmentů TCP

[1] klíč se sestává z kombinace zdrojové adresy, zdrojového portu, cílové adresy a cílového portu

Procesy:

(I) vytvoření segmentu a odložení zpracování zprávy, segment obsahuje tato data:

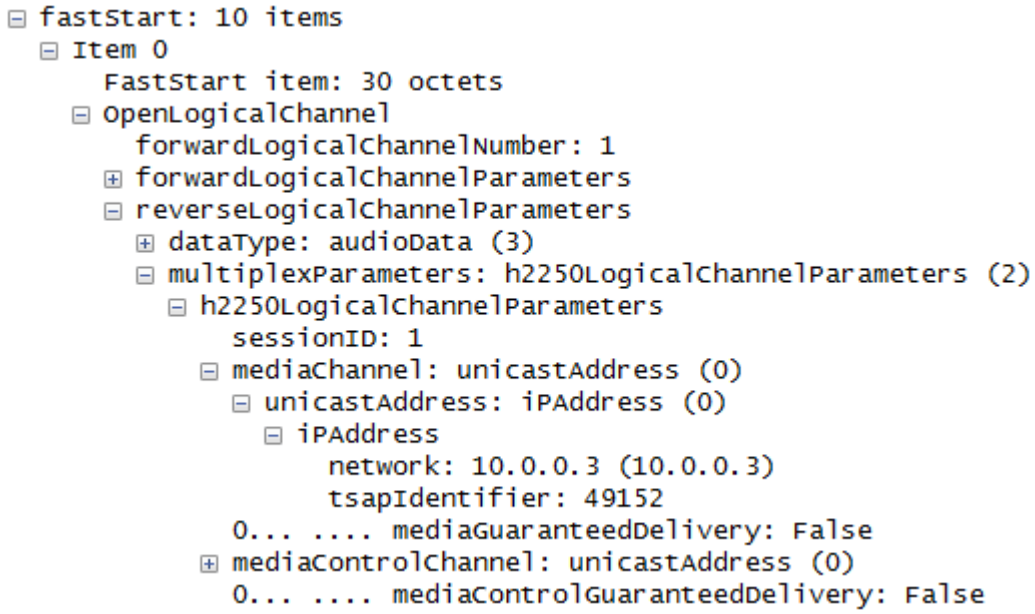
- dosažená délka obsahu
- očekávaná délka obsahu (získáno z TPKT)
- očekávané sekvenční číslo TCP
- obsah

(II) zpracování segmentu:

- navýšení dosažené délky o délku dat z nového paketu
- navýšení očekávaného sekvenčního čísla TCP o délku dat z nového paketu
- konkatenace původních dat a dat nových

(III) uložení aktualizovaného segmentu

(IV) zpracování dokončené zprávy



Obrázek 4.6: Ukázka části zprávy Setup

Na obrázku 4.6 je náhled části zprávy *Setup*, která je po dekódování uložena do struktury s informacemi o paket jako položka *reverse-ipAddress* s hodnotou *10.0.0.3:49152*.

Z různých zpráv jsou však získávány různé informace. Tabulka 4.3 ukazuje, které informace jsou vyhledávány ve kterých zprávách.

typ zprávy	Q.931 display, číslo volaného a volajícího	callIdentifier	FastStart – adresy RTP
Setup	✔	✔	✔
Call Proceeding	✔	✔	✔
Connect	✔	✔	✔
Alerting	✔	✔	✔
Information	✔	✔	✔
Release Complete	✔	✔	✘
Facility	✔	✔	✔
Progress	✔	✔	✔
zprávy **Empty**, **Status**, **Status Inquiry**, **Setup Acknowledge** a **Notify** nejsou použity			

Tabulka 4.3: Informace získávané z různých zpráv Call Setup

4.3.4 Správce hovorů

Správce hovorů zpracovává struktury s informacemi o paketech z dekodéru signalizačních zpráv, čímž vytváří, aktualizuje a ruší záznamy o hovorech. Záznam o hovoru je samostatný objekt, jeho struktura je uvedena v Tabulce 4.2.

Součástí zpracování hovorů je i úprava filtrovacích pravidel pro průchod paketů. Při detekci nového hovoru je přidáno pravidlo pro průchod příslušných paketů RTP, při získání nové informace může být toto pravidlo upraveno (například při získání další adresy proudu RTP) a při detekci ukončení hovoru je toto pravidlo odebráno.

Dále správce hovorů rovněž zpracovává pakety RTP. Vyhledává příslušné hovory a zapisuje obsah těchto paketů do odpovídajících výstupních souborů (pro každý směr jeden soubor). Protože neobsahují pakety RTP informace o tom, ze kterého hovoru pochází, je nutné vyhledávání provádět opačně. A to tak, že jsou postupně procházeny všechny hovory a jsou porovnávány položky s informacemi o adresách toků RTP se skutečnými adresami příchozích paketů. Po nalezení rodičovského hovoru je obsah paketu zapsán do příslušného výstupního souboru.

Tabulka 4.4 zobrazuje seznam podporovaných kodeků. Některé kodeky jsou navíc převáděny do formátu OGG, který lze přehrát v běžném přehrávači multimédií.

kodek	zpracování	konverze
A-law 64kb, 56kb	✔	✔
μ-law 64kb, 56kb	✔	✔
GSM 6.10	✔	✘
iLBC 30ms	✔	✘
iLBC 20ms	✔	✘
G.722 64kb, 56kb, 48kb	✔	✘
G.7231	✔	✘
G.728	✔	✘
G.729	✔	✘

Tabulka 4.4: Podporované kodeky

4.3.5 XML exportér

Tato komponenta transformuje databázi správce hovorů s metadaty o hovorech do souboru XML, který může použit pro další zpracování. Do XML exportéru jsou ukládány objekty s informacemi o již ukončených hovorech a při ukončení programu jsou navíc získána data o dosud neukončených hovorech a dochází k samotnému exportu. Ukázka exportu je na Obrázku 4.3.

4.4 Prostředí a knihovny

Jako implementační jazyk byl zvolen Python, neboť je to jazyk vhodný pro rychlý návrh, obsahuje knihovny pro přístup k síťové vrstvě i zpracování samotných paketů. Pro spuštění programu je nutné mít nainstalovaný Python verze 2.6.

Pro přístup k síťové vrstvě byla využita knihovna pcapy, která umožňuje zachytávat jednak pakety ze síťového rozhraní (online či live zachytávání a zpracování) a také umožňuje zpracovávat pakety již zachycené a uložené jako pcap soubor (offline zpracování).

Knihovna impacket byla využita pro zpracování paketů do úrovně transportní vrstvy.

4.5 Ovládání programu

Jelikož se jedná o konzolovou aplikaci, ovládání programu je řešeno parametry příkazové řádky.

V unixovém systému s nainstalovaným Pythonem 2.6 lze program spustit následujícím příkazem:

```
$ python voip_h323_sniffer.py [--mode=MODE] --input=DEVICE
```

V systémech Windows je příkaz následující:

```
> PATH_TO_PYTHON\python.exe voip_h323_sniffer.py [--mode=MODE] --input=DEVICE
```

Pokud je v systému Windows správně nastavena asociace podle koncovky souboru, stačí příkaz:

```
> voip_h323_sniffer.py [--mode=MODE] --input=DEVICE
```

Význam a možnosti parametrů jsou následující:

MODE – režim běhu programu, může nabývat pouze dvou hodnot

- **offline** - analýza z pcap souboru (výchozí)
- **online** - analýza ze síťového rozhraní

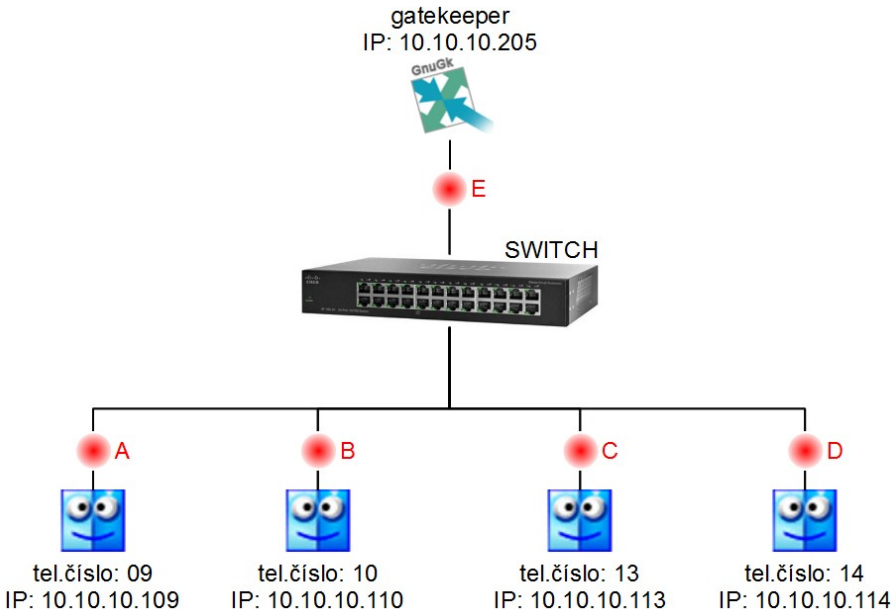
DEVICE – vstupní pcap soubor pro **offline** analýzu, nebo název rozhraní pro **online** analýzu

5 Testování

Tato kapitola popisuje testování vytvořeného programu. Testování budeme provádět s různými vstupními daty. Nejprve použijeme data, která jsme vytvořili v laboratoři a u kterých přesně víme, co obsahují a jaký by jím tedy měl odpovídat výstup. V druhé části budeme testovat na datech z reálného provozu páteřní sítě VUT. U takových dat nevíme, co obsahují, je však vhodné na nich naši aplikaci otestovat, neboť právě funkčnost na reálných datech je pro tuto aplikaci stěžejní. K porovnání výstupů bude využito programu Wireshark, který poskytuje dostatek informací o hovorech a jejich signalizaci.

5.1 Testování na laboratorních datech

Data vytvořená pro účely tohoto testu byla získána pomocí softwarových telefonů SJphone[4] a softwarové H.323 ústředny GnuGk[5]. K zachycení byl použit program Wireshark. Na obrázku 5.1 je zobrazena testovací architektura se čtyřmi SW telefony a jednou SW ústřednou, přes kterou se telefony registrují a vytvářejí hovory. Zachytávání probíhalo na stejných stanicích, na kterých byly spuštěny SW telefony a SW ústředna.



Obrázek 5.1: Testovací architektura

Body A až E jsou vyznačená místa, na kterých bylo prováděno zachytávání. Na daných bodech vznikly tyto soubory:

4 **http://www.sjlabs.com/sjp.html**
5 **http://www.gnugk.org/**

- A: **h09.pcap**
- B: **h10.pcap**
- C: **h13.pcap**
- D: **h14.pcap**
- E: **gk.pcap**

Soubory z bodů A až D (**h09** až **h14.pcap**) obsahuj signalizaci RAS s přihlášením a odhlášením příslušné stanice (**h09.pcap** obsahuje přihlášení stanice s telefonním číslem *09*, *h10* s telefonním číslem *10*, atd.). Tyto soubory obsahují také signalizaci hovorů CS a samotná hlasová data hovorů vedených z a na danou stanici. Hovory byly prováděny v tomto pořadí a směru:

1. stanice *h10* volá stanici *h13*
2. stanice *h09* volá stanici *h14*
3. stanice *h13* volá stanici *h14*
4. stanice *h09* volá stanici *h10*

Soubor s daty z bodu E obsahuje signalizaci RAS s přihlášením a odhlášením všech stanic, neboť tato signalizace probíhá mezi stanicemi a ústřednou. Signalizace CS a hlasová data v tomto souboru obsažena nejsou, neboť jsou vedena přímo mezi stanicemi.

Účelem tohoto testu je ověřit, že vytvořený nástroj korektně zpracovává signalizaci RAS i CS a že dokáže vytvořit odpovídající výstupy, totiž soubor XML a soubory s hlasovými daty typu RAW.

5.1.1 Průběh testů

Pro testování byly zvoleny dva soubory. Jednak soubor **h13.pcap** (odchyceno v bodě C), protože stanice *h13* hovor vytváří i přijímá, bude tedy možné ověřit funkcionalitu detekce hovorů v obou směrech. V druhém testu byl zvolen soubor **gk.pcap** (odchyceno v bodě E), protože v tomto bodě dochází je zachycena kompletní komunikace s ústřednou. Následuje příklad spuštění programu se souborem **h13.pcap** a terminálový výstup:

```
> voip_h323_sniffer.py --input=h13.pcap

RRQ from 10.10.10.113:1142, h323-id: h13, dialledDigits: 13, gateKeeper:
10.10.10.205:1719
RCF   to 10.10.10.113:1142, h323-id: h13, dialledDigits: 13, gateKeeper:
10.10.10.205:1719
call_0 detected
contents of call_0 stored at export/h13.pcap/call_0/
call_0 released
ACF destCallSignalAddress:10.10.10.114:1214 for user h13
(rasAddress:10.10.10.113:1141, callSignalAddress: 10.10.10.113:1142),
gateKeeper: 10.10.10.205:1719
call_1 detected
contents of call_1 stored at export/h13.pcap/call_1/
call_1 released
DCF for user h13(rasAddress:10.10.10.113:1141, callSignalAddress:
```

```
10.10.10.113:1142), gateKeeper: 10.10.10.205:1719
URQ from 10.10.10.113:1142, h323-id: h13, dialledDigits: 13, gateKeeper:
10.10.10.205:1719
UCF   to 10.10.10.113:1142, h323-id: h13, dialledDigits: 13, gateKeeper:
10.10.10.205:1719
number of calls processed: 2
```

Po spuštění vznikly v adresáři **`export/h13.pcap/`** následující soubory:

```
call_0\
  10.10.10.110;49152-10.10.10.113;49152_137fa540_GSM.raw
  10.10.10.113;49152-10.10.10.110;49152_028c4642_GSM.raw
  info.txt
call_1\
  10.10.10.113;49154-10.10.10.114;49154_2ec83252_G.711,A-law.raw
  10.10.10.114;49154-10.10.10.113;49154_0108897e_G.711,A-law.raw
  info.txt
export.xml
```

Z terminálového výstupu je vidět, že se stanice s IP adresou *10.10.10.113*, tedy stanice *h13* nejprve přihlásila (zpráva *RRQ*) na ústředně (IP adresa *10.10.10.205*) s uživatelským jménem *h13* a telefonním číslem *13*. Přihlášení bylo úspěšné a bylo ústřednou potvrzeno (zpráva *RCF*). Následně byl zachycen příchozí hovor, jehož obsah je ukládán do adresáře **`call_0/`**, který se nachází v adresáři pojmenovaném po vstupním souboru. Následuje potvrzení požadavku na spojení vyslaného stanicí *h13* (zpráva *ACF*) a bezprostředně poté je detekován odchozí hovor, jehož obsah je uložen do adresáře **`call_1/`**. Poté už následuje pouze potvrzení odpojení (zpráva *DCF*) a potvrzené odhlášení stanice *h13* (zprávy *URQ* a *UCF*).

Tabulka 5.1 zobrazuje události získané z testovacího souboru **`h13.pcap`** a informace o nich.

událost	**čas**	**další atributy**
přihlášení	28. června 2011, 12:13:28	id: h13, tel.číslo: 13, ústředna: 10.10.10.205:1719, stav: potvrzené
první hovor	28. června 2011, 12:20:15 28. června 2011, 12:21:00	zdroj: 10.10.10.110:49152, cíl: 10.10.10.113:49152, stream[0]: {kodek: GSM, směr: 10.10.10.110:49152 – 10.10.10.113:49152}, stream[1]: {kodek: GSM, směr: 10.10.10.113:49152 – 10.10.10.110:49152}
potvrzení spojení	28. června 2011, 12:21:21	ústředna: 10.10.10.205:1719, adresa volané strany: 10.10.10.114:1214
druhý hovor	28. června 2011, 12:21:21 28. června 2011, 12:22:02	zdroj: 10.10.10.113:49154, cíl: 10.10.10.114:49154, stream[0]: {kodek: G.711 A-law, směr: 10.10.10.114:49154 – 10.10.10.113:49154}, stream[1]: {kodek: G.711 A-law, směr: 10.10.10.113:49154 – 10.10.10.114:49154}
potvrzení rozpojení	28. června 2011, 12:22:02	ústředna: 10.10.10.205:1719
odhlášení	28. června 2011, 12:23:06	ústředna: 10.10.10.205:1719

Tabulka 5.1: Události získané ze souboru h13.pcap

Na obrázku 5.2 je pak zobrazen výstupní soubor XML vzniklý zpracováním naším nástrojem. Obsahuje informace o činnosti stanice *h13*, podobně jako terminálový výstup. Je však doplněn o další informace, které se v terminálovém výstupu nevyskytují (časové značky, informace o hovorech, atd.). Terminálový výstup je totiž pouze informativní zdroj, hlavním produktem je právě soubor ve formátu XML, který obsahuje všechny získané informace. Informace o hovorech jsou obsaženy i pro každý hovor zvlášť v jeho adresáři (**`call_0/`**, **`call_1/`**, atd.) v souboru **`info.txt`**. Soubor XML však obsahuje informace o všech hovorech z daného vstupního souboru.

Po spuštění nástroje se souborem **`gk.pcap`** byl získán následující terminálový výstup:

```
> voip_h323_sniffer.py --input=gk.pcap

RRQ from 10.10.10.113:1142, h323-id: h13, dialledDigits: 13, gateKeeper:
10.10.10.205:1719
RCF   to 10.10.10.113:1142, h323-id: h13, dialledDigits: 13, gateKeeper:
10.10.10.205:1719
RRQ from 10.10.10.109:1134, h323-id: h09, dialledDigits: 09, gateKeeper:
10.10.10.205:1719
RCF   to 10.10.10.109:1134, h323-id: h09, dialledDigits: 09, gateKeeper:
10.10.10.205:1719
RRQ from 10.10.10.110:1143, h323-id: h10, dialledDigits: 10, gateKeeper:
10.10.10.205:1719
RCF   to 10.10.10.110:1143, h323-id: h10, dialledDigits: 10, gateKeeper:
10.10.10.205:1719
RRQ from 10.10.10.114:1214, h323-id: h14, dialledDigits: 14, gateKeeper:
10.10.10.205:1719
RCF   to 10.10.10.114:1214, h323-id: h14, dialledDigits: 14, gateKeeper:
10.10.10.205:1719
ACF destCallSignalAddress:10.10.10.113:1142 for user h10
(rasAddress:10.10.10.110:1142, callSignalAddress: 10.10.10.110:1143),
gateKeeper: 10.10.10.205:1719
ACF destCallSignalAddress:10.10.10.114:1214 for user h09
(rasAddress:10.10.10.109:1132, callSignalAddress: 10.10.10.109:1134),
gateKeeper: 10.10.10.205:1719
DCF for user h09(rasAddress:10.10.10.109:1132, callSignalAddress:
10.10.10.109:1134), gateKeeper: 10.10.10.205:1719
DCF for user h10(rasAddress:10.10.10.110:1142, callSignalAddress:
10.10.10.110:1143), gateKeeper: 10.10.10.205:1719
ACF destCallSignalAddress:10.10.10.114:1214 for user h13
(rasAddress:10.10.10.113:1141, callSignalAddress: 10.10.10.113:1142),
gateKeeper: 10.10.10.205:1719
ACF destCallSignalAddress:10.10.10.110:1143 for user h09
(rasAddress:10.10.10.109:1132, callSignalAddress: 10.10.10.109:1134),
gateKeeper: 10.10.10.205:1719
DCF for user h13(rasAddress:10.10.10.113:1141, callSignalAddress:
10.10.10.113:1142), gateKeeper: 10.10.10.205:1719
DCF for user h09(rasAddress:10.10.10.109:1132, callSignalAddress:
10.10.10.109:1134), gateKeeper: 10.10.10.205:1719
URQ from 10.10.10.113:1142, h323-id: h13, dialledDigits: 13, gateKeeper:
10.10.10.205:1719
UCF   to 10.10.10.113:1142, h323-id: h13, dialledDigits: 13, gateKeeper:
10.10.10.205:1719
URQ from 10.10.10.114:1214, h323-id: h14, dialledDigits: 14, gateKeeper:
10.10.10.205:1719
UCF   to 10.10.10.114:1214, h323-id: h14, dialledDigits: 14, gateKeeper:
10.10.10.205:1719
URQ from 10.10.10.109:1134, h323-id: h09, dialledDigits: 09, gateKeeper:
10.10.10.205:1719
UCF   to 10.10.10.109:1134, h323-id: h09, dialledDigits: 09, gateKeeper:
```

```
10.10.10.205:1719
URQ from 10.10.10.110:1143, h323-id: h10, dialledDigits: 10, gateKeeper:
10.10.10.205:1719
UCF   to 10.10.10.110:1143, h323-id: h10, dialledDigits: 10, gateKeeper:
10.10.10.205:1719
number of calls processed: 0
```

Z terminálového výstupu je vidět, že se nejprve všechny stanice přihlásily a poté následovaly hovory v pořadí, které bylo uvedeno výše. Nakonec se všechny stanice odhlásily. Žádné hovory nebyly detekovány, neboť signalizace (i hlasová data) byla směrována přímo, mimo ústřednu.

V adresáři **`export/gk.pcap/`** tedy vznikl pouze jeden soubor:

```
export.xml
```

Vstupní soubory použité v tomto testu jsou poskytnuty na přiloženém CD v adresáři **`tests/`**. Vytvořené výstupní soubory z těchto testů jsou rovněž poskytnuty na přiloženém CD, v adresáři **`tests_results/`**. Testy je tedy možné kdykoliv zopakovat a porovnat výsledky.

```
<?xml version="1.0" encoding="UTF-8"?>
<log>
  <user guid="10.10.10.113:1142">
    <protocol val="H.323">
      <event state="confirmed" type="registration">
        <timestamp>Jun 28, 2011 12:13:28</timestamp>
        <h323-id>h13</h323-id>
        <dialledDigits>13</dialledDigits>
        <rasAddress>10.10.10.113:1141</rasAddress>
        <callSignalAddress>10.10.10.113:1142</callSignalAddress>
        <gateKeeper>10.10.10.205:1719</gateKeeper>
      </event>
      <event type="call">
        <callid>07c4d7dcc1c31246814bec2cabb55239</callid>
        <timestamp type="begin">Jun 28, 2011 12:20:15</timestamp>
        <timestamp type="end">Jun 28, 2011 12:21:00</timestamp>
        <src>10.10.10.110:49152</src>
        <dst>10.10.10.113:49152</dst>
        <content codec="GSM(3)" destination="10.10.10.113:49152" id="0"
          source="10.10.10.110:49152" type="RTP stream">
          export/h13.pcap/call_0/10.10.10.110;49152-10.10.10.113;49152....raw
        </content>
        <content codec="GSM(3)" destination="10.10.10.110:49152" id="1"
          source="10.10.10.113:49152" type="RTP stream">
          export/h13.pcap/call_0/10.10.10.113;49152-10.10.10.110;49152....raw
        </content>
      </event>
      <event state="confirmed" type="admission">
        <timestamp>Jun 28, 2011 12:21:21</timestamp>
        <gateKeeper>10.10.10.205:1719</gateKeeper>
        <destCallSignalAddress>10.10.10.114:1214</destCallSignalAddress>
      </event>
      <event type="call">
        <callid>474e1c349672a242a302689cf697ee99</callid>
        <timestamp type="begin">Jun 28, 2011 12:21:21</timestamp>
        <timestamp type="end">Jun 28, 2011 12:22:02</timestamp>
        <src>10.10.10.113:49154</src>
        <dst>10.10.10.114:49154</dst>
        <content codec="PCMA,A-law(8)" destination="10.10.10.114:49154" id="0"
          source="10.10.10.113:49154" type="RTP stream">
          export/h13.pcap/call_1/10.10.10.113;49154-10.10.10.114;49154....raw
        </content>
        <content codec="PCMA,A-law(8)" destination="10.10.10.113:49154" id="1"
        source="10.10.10.114:49154" type="RTP stream">
        export/h13.pcap/call_1/10.10.10.114;49154-10.10.10.113;49154....raw
      </content>
      </event>
      <event state="confirmed" type="disengage">
        <timestamp>Jun 28, 2011 12:22:02</timestamp>
        <gateKeeper>10.10.10.205:1719</gateKeeper>
      </event>
      <event state="confirmed" type="unregistration">
        <timestamp>Jun 28, 2011 12:23:06</timestamp>
        <gateKeeper>10.10.10.205:1719</gateKeeper>
      </event>
    </protocol>
  </user>
</log>
```

Obrázek 5.2: Výstupní soubor XML, test na laboratorních datech

5.2 Testování na datech z reálného provozu

Kromě testování na laboratorních byly prováděny testy s daty z reálného provozu – z páteřní sítě VUT. Dopředu tedy nevíme, co tato data obsahují a můžeme tedy lépe otestovat situace, které z nějakého důvodu nenastaly u dat z laboratoře – například využití jiných položek v signalizačních zprávách z důvodu použití jiných zařízení.

Výstup našeho nástroje je nutné porovnat s jiným referenčním nástrojem, k čemuž jsme využili program Wireshark, kde jsme sledovali počet detekovaných hovorů. Ke každému testovanému souboru bude uvedena jeho velikost a počet paketů, které obsahuje. Dále uvedeme počet detekovaných hovorů programem Wireshark a naším nástrojem a čas zpracování na počítači, na kterém bylo prováděno testování. Čas zpracování byl měřen pomocí příkazu *Measure-Command* v příkazové řádce *PowerShell*. Výsledky těchto testů jsou uvedeny v tabulce 5.2.

Testování bylo prováděno na následující konfiguraci:

CPU: Intel Core i5-2540M @ 2,6 GHz
RAM: 8 GB DDR3-1333 (667 MHz)
HDD: Intel SSDSA2BW160G3L (160 GB, SATA II)
OS: Windows 7 x64 Professional

soubor	velikost [B]	počet paketů	počet detekovaných hovorů		čas zpracování
			Wireshark	nástroj	
DUMP1.pcap	26 145 757	191 119	299	299	103,5 s
DUMP2.pcap	770 219	5 836	201	201	4,5 s
DUMP3.pcap	651 167 051	725 175	0	0	242 s

Tabulka 5.2: Výsledky testů na datech z reálného provozu

Z výsledků těchto testů je vidět, že náš nástroj dosahuje v detekci hovorů stejných výsledků, jako program Wireshark, což je uspokojivé.

Testovací data z těchto testů není možné poskytnout na CD, protože obsahují citlivé informace.

5.3 Zhodnocení testů

Provedli jsme dva typy testů, každý za jiným účelem. Nejprve jsme použili testovací data z laboratoře, která jsme vytvořili pomocí softwarových nástrojů. U těchto dat jsme přesně věděli, co obsahují a účelem těchto testů bylo ověřit, že vytvořený nástroj dokáže korektně zpracovat požadované události, signalizační zprávy a pakety s hlasovými daty. Díky těmto testům jsme dokázali, že vytvořený nástroj dokáže získat všechny podstatné informace o hovorech.

V druhé části jsme testovali na datech z páteřní sítě VUT, tedy na reálném provozu. Obsah těchto dat nebyl znám a proto bylo nutné výsledky porovnávat s jiným nástrojem, konkrétně s programem Wireshark. Účelem této sady testů bylo zjištění, zda má vytvořený nástroj srovnatelné schopnosti jako referenční program v oblasti detekce hovorů vedených přes H.323. Výsledkem těchto testů bylo zjištění, že vytvořený nástroj dokáže detekovat stejné množství hovorů, jako program Wireshark.

6 Závěr a zhodnocení

Cílem této bakalářské práce bylo seznámit se s architekturou IP telefonie postavené na signalizačních protokolech rodiny H.323, nastudovat transportní protokol RTP a prozkoumat možnosti nástrojů pro sledování a analýzu VoIP pomocí H.323. Následně bylo cílem navrhnout nástroj pro detekci a analýzu hlasových přenosů přes H.323.

V teoretické části práce jsme popsali jednotlivé součásti standardu H.323 a uvedli jsme možné způsoby ustavení hovoru, které jsou podstatné pro následnou analýzu. Následně jsme popsali nástroje použitelných k analýze hlasové komunikace přes H.323.

Největší pozornost jsme věnovali návrhu a implementaci nástroje, kde jsme popsali jeho jednotlivé komponenty. Také jsme uvedli jaké informace jsou důležité ke korektní analýze hlasové komunikace a jakým způsobem je vyhledat v síťovém provozu. Dále jsme popsali způsob vyhledání a zpracování hlasových či jiných multimediálních dat podle informací získaných ze signalizačních zpráv a způsob přiřazení těchto dat k jednotlivým hovorům.

Navržený nástroj jsme implementovali v jazyce Python s využitím knihoven *pcapy* a *impacket*. Podstatnou část práce jsme věnovali testování vytvořeného nástroje. Testovali jsme jednak na datech z laboratoře, kde jsme přesně věděli, co je obsahem těchto dat a výstup nástroje jsme mohli porovnat přímo s námi vytvořenými scénáři. Při testování s daty z reálného provozu, jehož účel bylo ověření nástroje v reálných podmínkách, jsme k získání referenčních výsledků využili program Wireshark.

Z hlediska dalšího vývoje nástroje by bylo vhodné rozšířit nástroj o vizualizační část a případně o funkcionalitu zpracovávající hovory signalizačního protokolu *SIP*.

Literatura

[1] ITU-T, Telecommunication standartization sector of ITU: *H.323 v7 (12/2009), Packet-based multimedia communications systems*.

[2] ITU-T, Telecommunication standartization sector of ITU: *X.680 (11/2008), Abstract Syntax Notation One (ASN.1): Specification of basic notation*.

[3] ITU-T, Telecommunication standartization sector of ITU: *X.691 (11/2008), Information technology – ASN.1 encoding rules: Specification of Packed Encoding Rules (PER)*.

[4] ITU-T, Telecommunication standartization sector of ITU: *H.225 (12/2009), Call signalling protocols and media stream packetization for packet-based multimedia communication systems*.

[5] Dubuisson, O.: *ASN.1 – Communication Between Heterogeneous Systems*, Waltham, Massachusetts, Morgan Kaufmann 2000, ISBN 01-26333-61-0.

[6] Pouffary, Y., Young, A.: *ISO Transport Service on top of TCP (ITOT)*, RFC 2126, 1997.

[7] ITU-T, Telecommunication standartization sector of ITU: *Q.931 (05/1998), ISDN user-network interface layer 3 specification for basic call control*.

[8] Schulzrinne, H., Casner, S., Frederick, R., Jacobson, V.: *RTP: A Transport Protocol for Real-Time Applications*, RFC 3550, 2003.

[9] Vozňák, M.: *Voice over IP*. Skripta VŠB-TU, Ostrava, 2009.

Seznam příloh

Příloha A: CD

Obsahuje zdrojové texty, manuál, testovací data z laboratoře a výsledky těchto testů.

src/	zdrojové texty
impacket/	modul impacket pro Python
libs*/	předkompilovaný modul pcapy (Linux a Windows)
src/	zdrojové moduly nástroje
voip_h323_sniffer_v2.py	hlavní modul nástroje
run.py	grafické rozhraní nástroje
tests/	testovací data
gk.pcap	
h09.pcap	
h10.pcap	
h13.pcap	
h14.pcap	
tests_results/	výsledky testů
gk.pcap/	
h09.pcap/	
h10.pcap/	
h13.pcap/	
h14.pcap/	
manual.pdf	manuál k ovládání nástroje

Příloha B: Manuál

Ovládání programu – příkazový řádek

Vlastní jádro nástroje lze spustit z příkazové řádky. Při *online* analýze jiná možnost ani není, grafické rozhraní je totiž určeno pro rychlou práci se soubory pcap při *offline* analýze.

V unixovém systému s nainstalovaným Pythonem 2.6 lze program spustit následujícím příkazem:

```
$ python voip_h323_sniffer.py [--mode=MODE] --input=DEVICE
```

V systémech Windows je příkaz následující:

```
> PATH_TO_PYTHON\python.exe voip_h323_sniffer.py [--mode=MODE] --input=DEVICE
```

Pokud je v systému Windows správně nastavena asociace podle koncovky souboru (.py), stačí příkaz:

```
> voip_h323_sniffer.py [--mode=MODE] --input=DEVICE
```

Význam a možnosti parametrů jsou následující:

MODE – režim běhu programu, může nabývat pouze dvou hodnot (pokud není přítomno vůbec, použije se výchozí režim)

- **offline** – analýza z pcap souboru (výchozí)
- **online** – analýza ze síťového rozhraní

DEVICE – vstupní pcap soubor pro *offline* analýzu, nebo název rozhraní pro *online* analýzu

Příklady spuštění programu

Spuštění *offline* analýzy nad souborem **`tests/h13.pcap`** (systém Windows):

```
> voip_h323_sniffer.py --input=tests/h13.pcap
```

A systém typu *nix:

```
$ python voip_h323_sniffer.py --input=tests/h13.pcap
```

Při spuštění v režimu *online* bez zadání vstupního zařízení program vypíše všechna dostupná zařízení. Příklad spuštění na systému Windows:

```
> voip_h323_sniffer.py --mode=online
available network devices:
```

```
  \Device\NPF_{99F6BD14-D827-4438-9C2D-9C78542A0234}
  \Device\NPF_{335BD612-33A4-473E-8ABC-9EB69FE67A97}
```

Příklad spuštění na systému *nix:

```
$ python voip_h323_sniffer.py --mode=online
available network devices:
  em0
  lo0
```

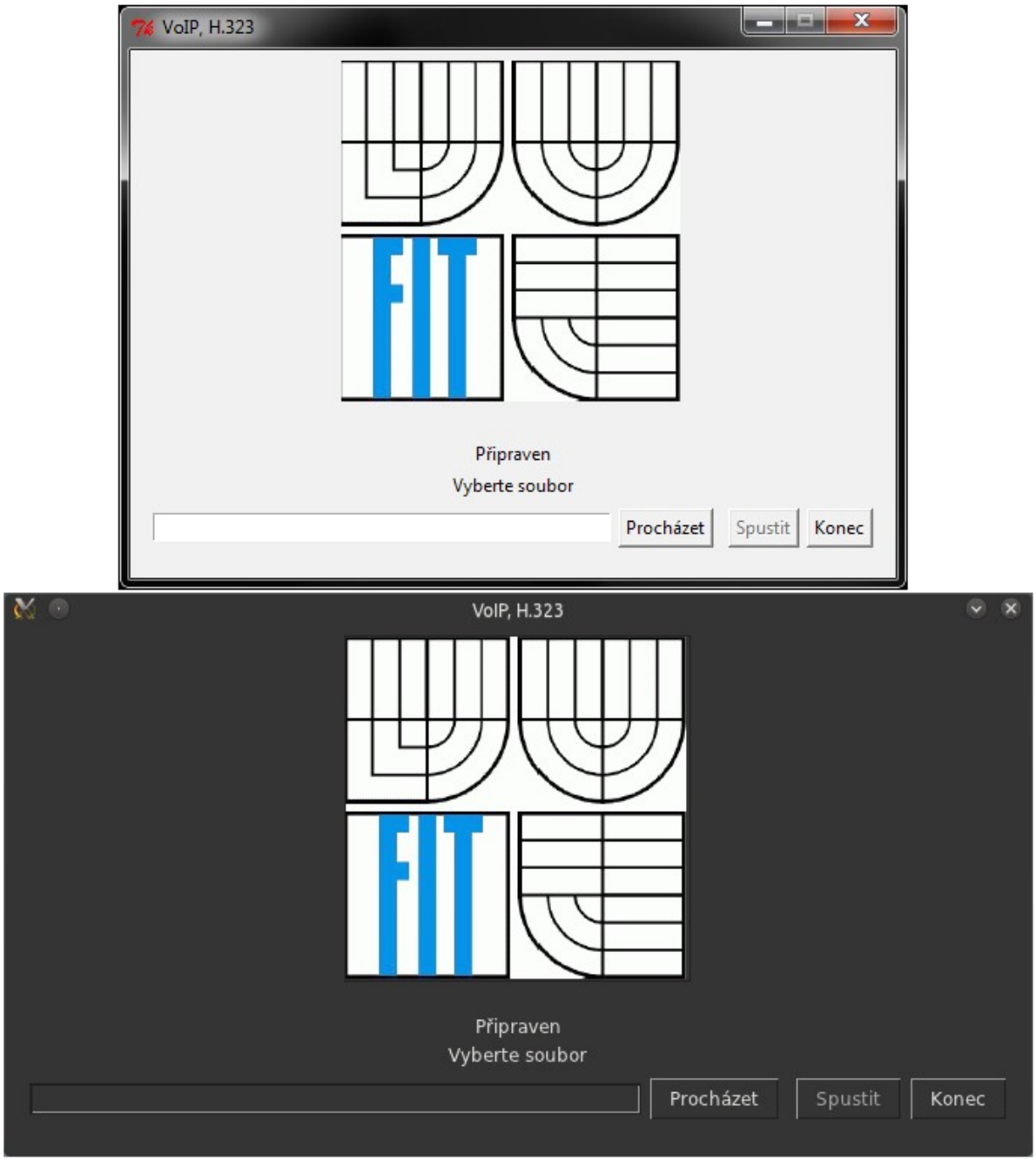
Při spuštění online analýzy na systému Windows je nutné zdvojit zpětná lomítka v názvu síťového zařízení. Příklad:

```
> voip_h323_sniffer_v2.py --mode=online --input=\\Device\\NPF_{99F6BD14-D827-
4438-9C2D-9C78542A0234}
```

Pro spuštění online analýzy na systémech *nix jsou kvůli přístupu k síťovým rozhraním vyžadována práva uživatele *root* (např. použitím příkazu *sudo*).

Grafické rozhraní

Grafické rozhraní je postaveno na modulu *Tkinter*, který bývá často součástí Pythonu samotného. Pokud však není, je třeba jej doinstalovat. GUI je velice jednoduché a slouží ke snadnějšímu spuštění *offline* analýzy. Po stisknutí tlačítka *Procházet* je možné otevřít soubor, který chce uživatel zpracovat. Následně, po stisknutí tlačítka *Spustit* dojde ke spuštění jádra nástroje, které zpracuje vybraný soubor. Po dokončení analýzy jsou uživateli nabídnuty výsledky ve formě tabulky v souboru *HTML*. GUI lze ukončit tlačítkem *Konec*. Celé rozhraní je zobrazeno na následujícím obrázku.



*Grafické rozhraní – v systému Windows (nahoře) a *nix (dole, distribuce Kubuntu)*